

Web Table Integration and Profiling for Knowledge Base Augmentation

Inauguraldissertation
zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften
der Universität Mannheim

vorgelegt von

Oliver Lehmberg
aus Berlin

Mannheim, 2019

Dekan: Dr. Bernd Lübcke, Universität Mannheim
Referent: Professor Dr. Christian Bizer, Universität Mannheim
Korreferent: Professor Dr. Heiko Paulheim, Universität Mannheim
Dritter Prüfer: Professor Dr. Stefan Dietze, Heinrich Heine Universität Düsseldorf

Tag der mündlichen Prüfung: 26.09.2019

Abstract

HTML tables on web pages (“*web tables*”) have been used successfully as a data source for several applications. They can be extracted from web pages on a large-scale, resulting in corpora of millions of web tables. But, until today only little is known about the general distribution of topics and specific types of data that are contained in the tables that can be found on the Web. But this knowledge is essential to understanding the potential application areas and topical coverage of web tables as a data source. Such knowledge can be obtained through the integration of web tables with a knowledge base, which enables the semantic interpretation of their content and allows for their topical profiling. In turn, the knowledge base can be augmented by adding new statements from the web tables. This is challenging, because the data volume and variety are much larger than in traditional data integration scenarios, in which only a small number of data sources is integrated.

The contributions of this thesis are methods for the integration of web tables with a knowledge base and the profiling of large-scale web table corpora through the application of these methods. For this profiling, two corpora of 147 million and 233 million web tables, respectively, are created and made publicly available. These corpora are two of only three that are openly available for research on web tables. Their data profile reveals that most web tables have only very few rows, with a median of 6 rows per web table, and between 35% and 52% of all columns contain non-textual values, such as numbers or dates. These two characteristics have been mostly ignored in the literature about web tables and are addressed by the methods presented in this thesis.

The first method, T2K Match, is an algorithm for semantic table interpretation that annotates web tables with classes, properties, and entities from a knowledge base. Other than most algorithms for these tasks, it is not limited to the annotation of columns that contain the names of entities. Its application to a large-scale web table corpus results in the most fine-grained topical data profile of web tables at the time of writing, but also reveals that small web tables cannot be processed with high quality. For such small web tables, a method that stitches them into larger tables is presented and shown to drastically improve the quality of the results.

The data profile further shows that the majority of the columns in the web tables, where classes and entities can be recognised, have no corresponding properties in the knowledge base. This makes them candidates for new properties that can be added to the knowledge base. The current methods for this task, however, suffer from the oversimplified assumption that web tables only contain binary relations. This results in the extraction of incomplete relations from the web tables as new properties and makes their correct interpretation impossible. To increase the completeness, a method is presented that generates additional data from the context of the web tables and synthesizes n-ary relations from all web tables of a web site. The application of this method to the second large-scale web table corpus shows that web tables contain a large number of n-ary relations. This means that the data contained in web tables is of higher complexity than previously assumed.

Zusammenfassung

HTML-Tabellen auf Webseiten (*“Web Tables”*) wurden erfolgreich als Datenquelle für mehrere Anwendungen verwendet. Sie können in großem Umfang aus Webseiten extrahiert werden, was zu Korpora von Millionen von Web Tables führt. Bis heute ist jedoch nur wenig über die allgemeine Verteilung von Themen und Datentypen bekannt, die in den Tabellen enthalten sind. Dieses Wissen ist jedoch unerlässlich um die potenziellen Anwendungsbereiche und die thematische Abdeckung von Web Tables als Datenquelle zu verstehen. Dieses Wissen kann durch die Integration von Web Tables mit einer Wissensbasis gewonnen werden, was die semantische Interpretation ihres Inhalts und eine thematische Profilerstellung ermöglicht. Die Wissensbasis kann wiederum durch Hinzufügen neuer Aussagen aus den Web Tables erweitert werden. Dies ist eine Herausforderung, da Datenvolumen und Vielfältigkeit viel größer sind als in herkömmlichen Datenintegrationsszenarien, in denen nur eine geringe Anzahl von Datenquellen integriert wird.

Die Beiträge dieser Arbeit sind Methoden zur Integration von Web Tables mit einer Wissensbasis und zur Profilerstellung von großen Korpora von Web Tables. Für diese Profilerstellung werden zwei Korpora mit 147 Mio. bzw. 233 Mio. Web Tables erstellt und öffentlich zugänglich gemacht. Bei diesen Korpora handelt es sich um zwei von nur drei Korpora, die frei für die Erforschung von Web Tables zur Verfügung stehen. Aus ihnen geht hervor, dass die meisten Web Tables nur wenige Zeilen haben, mit einem Median von 6, und zwischen 35% und 52% aller Spalten nicht-textuelle Werte wie Zahlen oder Datumsangaben enthalten. Diese beiden Eigenschaften wurden in der Literatur zumeist ignoriert und werden von den vorgestellten Methoden adressiert. Die erste Methode, T2K Match, ist ein Algorithmus zur semantischen Interpretation von Tabellen, der Web Tables mit Klassen, Eigenschaften und Entitäten aus einer Wissensbasis annotiert. Anders als viele vergleichbare Algorithmen ist er nicht auf die Annotation von Spalten mit benannten Entitäten beschränkt. Seine Anwendung auf einen großen Korpus führt zum aktuell detailliertesten Datenprofil von Web Tables, zeigt jedoch auch, dass kleine Web Tables nicht mit hoher Qualität verarbeitet werden können. Für solche kleinen Web Tables wird eine Methode vorgestellt, mit der sie zu größeren Tabellen zusammengefügt werden, was die Qualität der Ergebnisse drastisch verbessert.

Das Datenprofil zeigt weiterhin, dass die Mehrheit der Spalten in Web Tables, in denen Klassen und Entitäten erkannt werden, keine entsprechenden Eigenschaften in der Wissensbasis hat. Dies macht sie zu Kandidaten für neue Eigenschaften. Die aktuellen Methoden für diese Aufgabe leiden unter der vereinfachten Annahme, dass Web Tables nur binäre Relationen enthalten. Dies führt zur Extraktion unvollständiger Relationen und macht deren korrekte Interpretation unmöglich. Um die Vollständigkeit zu erhöhen wird eine Methode vorgestellt, die zusätzliche Daten aus dem Kontext der Web Tables generiert und n-äre Relationen aus allen Seiten einer Website synthetisiert. Ihre Anwendung auf den zweiten Korpus zeigt, dass Web Tables viele n-äre Relationen enthalten. Dies bedeutet, dass die in Web Tables enthaltenen Daten komplexer sind als bisher angenommen.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Contributions	4
1.3	Thesis Outline	6
1.4	Published Work	8
I	Foundations	11
2	Data Profiling and Integration	13
2.1	Introduction	13
2.2	Preliminaries: Relational Data Model	13
2.3	Data Profiling	15
2.3.1	Single Source	16
2.3.2	Multiple Sources	17
2.4	Data Integration	18
2.4.1	Matching Tasks	19
2.4.2	Similarity Calculation	21
2.4.3	Matcher Architecture	24
2.4.4	Data Fusion	27
2.5	Conclusion	30
3	Knowledge Bases	31
3.1	Introduction	31
3.2	Preliminaries: RDF Data Model	32
3.3	Overview of Common Knowledge Bases	34
3.4	Knowledge Base Augmentation	40
3.5	Conclusion	45
4	Web Tables	47
4.1	Introduction	47
4.2	Related Work	49
4.3	The Web Table Extraction Process	51
4.3.1	Table Classification	52

4.3.2	Header Row Detection	57
4.3.3	Data Type Detection	59
4.3.4	Subject Column Detection	61
4.4	Web Table Corpora	62
4.4.1	Web Data Commons Web Tables Corpus 2012	64
4.4.2	Web Data Commons Web Tables Corpus 2015	68
4.5	Conclusion	74
II	Matching Web Tables to a Knowledge Base	77
5	Semantic Table Interpretation	79
5.1	Introduction	79
5.2	Related Work	81
5.2.1	Semantic Table Interpretation Tasks	81
5.2.2	Approaches	86
5.2.3	Comparability	92
5.2.4	Real World Applicability	94
5.3	T2D Gold Standard	95
5.3.1	Annotated Semantic Table Interpretation Tasks	95
5.3.2	Annotation Procedure	99
5.4	Method: T2K Match	102
5.4.1	Pre-processing	102
5.4.2	Matcher Workflow	104
5.5	Experiments	111
5.5.1	Parameter Optimisation	111
5.5.2	Matcher Evaluation	112
5.5.3	Comparison with other Approaches	115
5.6	Conclusion	117
6	Corpus Profiling	119
6.1	Introduction	119
6.2	Related Work	121
6.3	A Topical Profile for Web Tables	123
6.3.1	Methods	123
6.3.2	Correspondence Statistics	124
6.4	Quality of Extracted Triples	133
6.4.1	Methods	133
6.4.2	Fusion Results	138
6.5	Conclusion	143

7	Table Stitching	145
7.1	Introduction	145
7.2	Related Work	147
7.3	Method: Table Stitching	151
7.3.1	Union Tables	152
7.3.2	Stitched Union Tables	155
7.4	Improving Semantic Table Interpretation	158
7.4.1	Intra-Site Schema Matching	160
7.4.2	Evaluation on Individual Web Sites	162
7.4.3	Evaluation on a Random Sample	165
7.4.4	Result Analysis	168
7.5	Conclusion	172
III	Extending the Schema of a Knowledge Base	175
8	Schema Extension	177
8.1	Introduction	177
8.2	Related Work	179
8.3	Attribute Ranking	184
8.3.1	Attribute Matching	184
8.3.2	Ranking Strategies	186
8.3.3	Evaluation	187
8.3.4	Result Analysis	194
8.4	Attribute Categorisation	196
8.4.1	Categorisation Scheme	197
8.4.2	Manual Category Profiling	200
8.4.3	Classification Model	202
8.4.4	Corpus Profiling	206
8.5	Conclusion	208
9	Synthesizing N-ary Relations	211
9.1	Introduction	211
9.2	Related Work	214
9.3	Method: SNoW	216
9.3.1	Method Overview	217
9.3.2	Context Extraction	218
9.3.3	Schema Matching	219
9.3.4	Schema Integration	222
9.3.5	Functional Dependency Discovery	224
9.3.6	Schema Normalisation	228
9.4	Evaluation	231
9.4.1	Evaluation Datasets	231
9.4.2	Schema Matching	234

9.4.3	Relation Synthesis	234
9.5	Corpus Profiling	239
9.5.1	Topical Profile and Arity of Relations	240
9.5.2	Key Attribute Profiling	243
9.5.3	Cross-Site Schema Matching.	246
9.6	Conclusion	250
10	Conclusion	253
10.1	Summary	253
10.2	Research Impact	256
10.3	Limitations & Future Work	257
	List of Figures	259
	List of Tables	263
	Bibliography	265

Chapter 1

Introduction

1.1 Motivation

The Web is the largest source of information that is available to humankind, with billions of interlinked web pages, which are created and maintained by millions of people from all over the world. Everybody with access to the Internet can create content on the Web, which drives its constant growth and unparalleled diversity. Besides its world-wide accessibility, the success of the Web is due to the possibility of interlinking its content with hyperlinks. Such hyperlinks allow users to navigate to related content and transform isolated documents into a highly connected network [Meusel et al., 2015]. These connections, and the fact that all content on the Web is machine-readable, make it possible to automatically collect vast amounts of web pages, which enables empirical research about data and their representation as well as the creation of services such as search engines.

While the content on the Web is always machine-readable, it is in most cases not machine-understandable. Information on the Web is designed to be consumed by humans, and hence encoded in natural languages or audio-visual contents. This raw data can be stored, indexed, and retrieved by machines, but does not allow for higher-level applications, such as answering the questions “*which airline uses the code ‘NH’?*” or “*what was the average annual wage of a computer scientist in California in 2016?*”. To enable such applications, it is necessary to extract statements about the world from the data and integrate them with a knowledge base, which contains additional knowledge about the world such as “*an airline is a company*” or “*airlines have a property called code*”.

The need for such structured knowledge has been, and still is, a driving force for the creation and augmentation of knowledge bases: Large collections of facts which are organised in an ontology, i.e., a machine readable definition of a model of the real-world [Ehrlinger and Wöß, 2016], enabling machines to answer complex queries and perform inference. Today, knowledge bases such as DBpedia [Lehmann et al., 2015], Yago [Suchanek et al., 2007], Wikidata [Vrandečić and Krötzsch, 2014] or the Google Knowledge Graph [Singhal, 2012] contain mil-

NAME	WEBSITE	HUB	AIRLINE CODE
Aeroflot	www.aeroflot.ru/eng/	Moscow	SU
Air France	www.airfrance.fr	Paris	AF
All Nippon Airways	www.ana.co.jp	Tokyo	NH
Asiana Airlines	www.flyasiana.com/gateway/	Seoul	OZ
Cathay Pacific	www.cathaypacific.com	Hong Kong	CX
China Airlines	www.china-airlines.com	Taipei	CI
China Southern Airlines	www.cs-air.com	Guangzhou	CZ
Japan Airlines	www.jal.co.jp	Tokyo	JL

Figure 1.1: A web table listing airlines and their attributes.

lions of entities and billions of statements about these entities and their properties, yet they are still considered vastly incomplete. Such knowledge bases are created through the automated extraction of information from regularly structured sources on the Web, such as Wikipedia, and are then continuously augmented by integrating additional data sources. This augmentation of knowledge bases is the focus of this thesis, with respect to a particular form of data representation on the Web: *web tables*.

Besides unstructured natural language text and structured semantic annotations, which can be placed in the mark-up code by web site authors, web tables are a form of semi-structured data on the Web [Lim and Ng, 1999]. A web table is a table on a web page in which data are organised in rows and columns, but no explicit statements about the meaning of the data are available. An example is shown in Figure 1.1, which displays a web table with four columns and nine rows. For a human, it is easy to interpret that the table is about airlines and their properties, even if only a few parts of the table are understood. The basic structuring in rows and columns allows for an interpretation of the data, even without fully understanding its contents: All values in the same column and all values in the same row share some common semantics. Once these semantics are understood, for example by comparing the contents of a column with known values from a knowledge base, all its data, be it known or unknown, can be interpreted and integrated into the knowledge base.

The integration into a knowledge base enables applications to access the information contained in the web tables without requiring specific knowledge about the structure and context of the original source. The integration hence makes the information more accessible and usable by a broader range of consumers. Storing this information in a knowledge base rather than in a traditional database is beneficial because the data are not transformed into an application-specific schema.

Computer and Information Research Scientists

Conduct research into fundamental computer and information science as theorists, designers, or inventors. Develop solutions to problems in the field of computer hardware and software.

State	Employment (1)	Employment per thousand jobs	Location quotient (9)	Hourly mean wage	Annual mean wage (2)
California	4,950	0.31	1.64	\$60.39	\$125,620
Virginia	2,550	0.68	3.59	\$60.96	\$126,800
Maryland	2,490	0.94	4.99	\$54.38	\$113,110
Texas	1,810	0.15	0.82	\$47.52	\$98,830
New Jersey	1,530	0.39	2.04	\$60.52	\$125,880

Figure 1.2: A web table showing employment and wage statistics for different states and a specific profession, which is stated outside of the web table.

Instead, they are stored in a more general format that expresses information as subject-predicate-object triples, such as “*Germany hasCapital Berlin*”, and allows applications to access the information that they require through querying and reasoning.

The challenges that are encountered in this endeavour stem from the open nature of the Web: Everybody can put content on the Web, and this content stays on the Web if not explicitly removed, it might be replicated multiple times as it is copied, and it is not subjected to any quality control mechanism. This means that data on the Web can be incorrect, outdated, fictitious, encoded in a variety of different forms and languages or simply be irrelevant. Further, web pages are designed for humans and contain, besides the actual content, navigational elements, layout elements and advertisement. The challenge for knowledge base augmentation based on web data is hence to identify relevant content, recognise different possible representations of the same piece of information, understand its context, and to decide which sources to trust.

The current state of the art for the interpretation and integration of web table data with other data sources, such as knowledge bases, addresses these challenges with a variety of approaches, but is limited through simplified assumptions and generally results in data of rather low quality. The data profile of the two large-scale web table corpora that will be presented in this thesis shows that the majority of all web tables is very small and contains a large fraction of columns with non-textual content, such as numbers or dates. Many methods, however, focus exclusively on web table columns that contain the names of entities and hence ignore a large amount of the data that is contained in the web tables. The evaluation datasets that are used to evaluate these approaches are usually selected by searching for a set of known entity names, which leads to datasets that contain larger web tables than would be expected from a random sample. This leads to an overly optimistic evaluation of the approaches, as very small web tables, which are harder to interpret, are underrepresented. Finally, almost all approaches apply the assumption that there

exist only binary relations between the columns in web tables, which means that they fail for web tables that convey more complex information, such as the web table shown in Figure 1.2. This table presents employment and wage statistics in different states for computer and information research scientists, but this specific profession is not contained in the web table itself. A binary relation, for example between the first two columns, does not contain this information and is hence incomplete. This thesis will show that these limitations have a strong impact on the quality of the existing approaches through several data profiles of large-scale web table corpora and will present methods that overcome these limitations.

1.2 Contributions

This section summarises the contributions of this thesis and outlines how the state of the art is improved by these contributions. The contributions are made in three areas: (1) the public availability and profiling of large-scale corpora of web tables, (2) methods for semantic table interpretation and topical profiling of large scale corpora, and (3) methods for schema extension and topical and structural profiling of relations in web tables. The specific contributions in these areas are the following:

1. **Publication & Profiling of Web Table Corpora:** Before the work on this thesis, researchers repeatedly extracted privately held corpora, for which no detailed data profile was published and which could not be re-used by others. The contributions of this thesis are the publication of the large-scale Web Data Commons Web Tables Corpus (“WTC”) 2012 and 2015 as well as the creation and publication of data profiles of these corpora. At the time of writing, these two corpora and the Dresden Web Tables Corpus [Eberius et al., 2015] are the only publicly available large-scale web table corpora.

This contribution is joint work with Dominique Ritze and previously published in [Lehmberg et al., 2016]. I contributed to the creation of the data profiles and the publication of the corpora.

2. **T2D Gold Standard and T2K Match:** The comparability of methods for semantic table interpretation, i.e., the annotation of web tables with elements from a knowledge base, is limited through the inaccessibility of the methods and datasets that are used in the literature. The publication of the open-source T2K Match algorithm and the T2D gold standard for semantic table interpretation contributes to improving the comparability in this area.

This contribution is joint work with Dominique Ritze and previously published in [Ritze et al., 2015]. I developed the annotation tool, contributed to the annotations for the gold standard, to the design and implementation of the matching algorithm as well as to the experimental evaluation.

3. **Topical Profiling for Web Tables:** The only previous work to profile the topical content of web tables [Hassanzadeh et al., 2015] is limited to a class-level profile of large web tables (≥ 20 rows). The contribution of this thesis is the most fine-grained topical profile of a large-scale web table corpus at the time of writing. The profile analyses the WTC 2012 with respect to classes, entities, and properties from the DBpedia knowledge base as well as the quality of statements that can be extracted for knowledge base augmentation.

This contribution is joint work with Dominique Ritze and Yaser Oulabi and previously published in [Ritze et al., 2016]. I contributed to the implementation and execution of the large-scale matching, to the profiling of the results as well as to the evaluation of the data fusion results.

4. **Matching of Small Web Tables:** Very small web tables are the majority in the public corpora, but most approaches for semantic table interpretation and their evaluation ignore this, which leads to worse results than assumed from the literature. The contribution of this thesis is an experimental verification of this claim and a method to stitch (combine) small web tables, which drastically improves the quality of existing methods.
5. **Categorisation of Table Columns:** Most methods assume that web tables only contain binary relations, i.e., that all values can be understood in combination with the subject column of the web table. A manual profiling shows that this is often not the case and leads to the extraction of un-interpretable triples. Based on this profile, a categorisation system for different types of columns is proposed. The finding is supported by a large-scale profiling for these categories using a supervised classification model.
6. **Extraction of N-ary Relations:** Current methods can only extract binary relations or specific n-ary relations that include time information from web tables. The contribution of this thesis is the first method for the extraction of general n-ary relations from web tables and a data profile that shows that such relations are frequent among web tables and contain much more complex information than previously assumed.
7. **Open-Source Data Integration Methods & Public Evaluation Datasets:** During the work on this thesis, multiple data integration methods as well as a general framework for data integration [Lehmberg et al., 2017] were developed and published as open source.¹²³⁴ Further, evaluation datasets for semantic table interpretation (Chapter 5, 7), web table matching (Chapter 7, 9), and schema extension (Chapter 9) were created and made publicly available. This ensures the reproducibility of the experiments and enables the comparison of new methods with the presented results.

¹<https://github.com/olehmberg/winter>

²<https://github.com/olehmberg/T2KMatch>

³<https://github.com/olehmberg/WebTableStitching>

⁴<https://github.com/olehmberg/snow>

1.3 Thesis Outline

This section provides a summary of each of the following chapters and their contributions. The thesis consists of three parts. The first part introduces the methodological foundations of data profiling and data integration as well as an introduction of knowledge bases and the data source that is used throughout the thesis: web tables. The second part describes methods and data profiling results for semantic table interpretation and knowledge base augmentation with respect to the slot-filling task. The third and final part presents methods and data profiling results for the schema-extension task.

PART I: Foundations

The first part of this thesis introduces the methods and the data sources which are used in all following chapters. This comprises the foundations of data profiling and integration, which are the basis of all methods described in the following chapters, an introduction to knowledge bases, which are used to gain a semantic interpretation of web tables and are the target of augmentation, as well as an introduction to web tables, which are the data source for all discussed tasks.

Chapter 2: Data Profiling and Integration. This chapter presents the data profiling and integration methods, which form the foundation of all methods presented in this thesis. In particular, the data integration process and methods for schema and data matching as well as data fusion are introduced.

Chapter 3: Knowledge Bases. This chapter introduces common knowledge bases used in the research literature and introduces the different knowledge base augmentation tasks. A special focus is on the DBpedia knowledge base, which is used in all experiments in this thesis as the target knowledge base.

Chapter 4: Web Tables. This chapter introduces web tables and their characteristics, and surveys methods used to extract web tables from web pages and understand their structure. Further, the extraction methodology and the data profile of two large-scale corpora, the Web Data Commons Web Tables Corpus 2012 and 2015, are presented and compared to similar corpora.

PART II: Matching Web Tables to a Knowledge Base

The second part of this thesis describes methods that match web tables to a knowledge base and reports the results of applying these methods to a large-scale corpus of web tables. These results represent the most fine-grained topical profile of web tables at the time of writing and reveal shortcomings of the semantic table interpretation methods, leading to the development of a method that drastically improves their performance.

Chapter 5: Semantic Table Interpretation. This chapter surveys the state of the art in semantic table interpretation and presents a new algorithm for this area, which iteratively solves the schema and data matching tasks that are necessary to create triples from web tables that can be used for knowledge base augmentation. The proposed algorithm, T2K Match, annotates individual web tables with classes, their columns with properties, and their rows with entities from a target knowledge base. Further, the lack of comparability of existing methods is addressed by the creation of the new T2D gold standard for semantic table interpretation, which is made publicly available and has already been used by other authors to compare their approaches.

Chapter 6: Corpus Profiling. This chapter presents a topical profile of the Web Data Commons Web Tables Corpus 2012 that is created through the application of the T2K Match algorithm. The profile details the occurrences of classes, properties, and entities from the DBpedia knowledge base in the web tables, and represents the most fine-grained topical profile of a large-scale web table corpus at the time of writing. Further, the truthfulness of the extracted triples is estimated as an additional profiling dimension and several data fusion methods for the selection of high-quality triples for knowledge base augmentation are evaluated.

Chapter 7: Table Stitching. This chapter presents experimental evidence that the commonly used evaluation datasets for semantic table interpretation are biased towards large web tables and result in overly optimistic evaluations. To improve the matching performance, a method is presented that stitches (combines) the web tables of a web site before the application of an existing matching algorithm. The result analysis shows that this method drastically improves the matching performance and is especially effective for small web tables, which are the majority in the publicly available web table corpora.

PART III: Extending the Schema of a Knowledge Base

The third part of this thesis presents methods for schema extension and revises oversimplified assumptions that are made by most approaches for web tables. An initial profiling of candidates for schema extension shows that many columns in web tables, which do not have corresponding properties in the knowledge base, are in non-binary relations, which cannot be handled by most current methods. Hence, a new method is designed that can extract such n-ary relations from web tables. The data profiling of the extracted relations shows that web tables contain much more complex information than previously assumed.

Chapter 8: Schema Extension. This chapter presents a comparative evaluation of several methods for schema extensions on the WTC 2012. The evaluation indicates that not all columns in web tables can be interpreted without considering additional data. It is argued that the usually employed criterion of relevance is

insufficient to guarantee the extraction of triples that can be inserted into the knowledge base, and that completeness must also be taken into account. Consequently, a classification scheme for column in web tables is introduced that categorises them according to their relationship to the subject column of the web table. The design and application of a supervised classifier for these categories verifies that the majority of the columns in web tables are, in contrast to the assumptions of state-of-the-art methods, in a non-binary relation with the subject column.

Chapter 9: Synthesizing N-ary Relations. This chapter presents a method that identifies which columns in a web table form a relation that can be used for schema extension, addressing the problem of un-interpretable triples which are created by state-of-the-art methods. This is possible through the stitching of all web tables from the same web site and the extraction of additional context data from the web pages that contain the web tables. The chapter further presents a profile of the WTC 2015 that shows that a large fraction of the columns in the web tables are in non-binary relations, and that these non-binary relations often require context data which do not exist in the original web tables to be interpretable.

Chapter 10: Conclusion. This chapter summarises the contributions of this thesis. It further presents its research impact through the discussion of research publications that have used the datasets and methods that are described in this thesis, and finally discusses limitations and directions for future work.

1.4 Published Work

The work presented in this thesis has been published previously in international journals and proceedings of international conferences and workshops:

Web Table Corpora:

- **[Lehmberg et al., 2016]** Lehmberg, O., Ritze, D., Meusel, R., & Bizer, C. (2016). A Large Public Corpus of Web Tables containing Time and Context Metadata. *In Proceedings of the 25th International Conference Companion on World Wide Web* (pp. 75-76).

Semantic Table Understanding & Knowledge Base Augmentation:

- **[Ritze et al., 2015]** Ritze, D., Lehmberg, O., & Bizer, C. (2015). Matching HTML tables to DBpedia. *In Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics* (p. 10).
- **[Ritze et al., 2016]** Ritze, D., Lehmberg, O., Oulabi, Y., & Bizer, C. (2016). Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. *In Proceedings of the 25th international conference on world wide web* (pp. 251-261).

- **[Lehmberg and Bizer, 2016]** Lehmberg, O., & Bizer, C. (2016). Web table Column Categorisation and Profiling. *In Proceedings of the 19th International Workshop on Web and Databases (p. 4).*
- **[Lehmberg and Hassanzadeh, 2018]** Lehmberg, O., & Hassanzadeh, O. (2018). Ontology Augmentation Through Matching with Web Tables. *In Ontology Matching: OM-2018: Proceedings of the ISWC Workshop (p. 37).*

Relation Synthesis from Web Tables:

- **[Lehmberg et al., 2015]** Lehmberg, O., Ritze, D., Ristoski, P., Meusel, R., Paulheim, H., & Bizer, C. (2015). The Mannheim Search Join Engine. *In Web Semantics: Science, Services and Agents on the World Wide Web, 35, 159-166.*
- **[Lehmberg and Bizer, 2017]** Lehmberg, O., & Bizer, C. (2017). Stitching web tables for improving matching quality. *In Proceedings of the VLDB Endowment, 10(11), 1502-1513.*
- **[Lehmberg and Bizer, 2019b]** Lehmberg, O., & Bizer, C. (2019). Synthesizing N-ary Relations from Web Tables. *In Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics (Article 17, 12 pages).*
- **[Lehmberg and Bizer, 2019a]** Lehmberg, O., & Bizer, C. (2019). Profiling the Semantics of N-ary Web Table Data. *In Proceedings of the International Workshop on Semantic Big Data (Article 5, 6 pages).*

Open-Source Data Integration Software:

- **[Lehmberg et al., 2017]** Lehmberg, O., Brinkmann, A., & Bizer, C. (2017). WInte.r - A Web Data Integration Framework. *In International Semantic Web Conference (Posters, Demos & Industry Tracks).*

Part I

Foundations

Chapter 2

Data Profiling and Integration

2.1 Introduction

In today's world, information systems and their applications are pervasive and have a tremendous impact on businesses and individuals alike. The strength of the used systems lies in their capability of processing large amounts of data efficiently and they can hence assist their users to achieve their goals, be it the execution of a business process or finding the cheapest price for a consumer product. To enable these applications, however, the systems need access to data which is prepared in a machine-understandable format. This preparation of data is the focus of the research areas of data profiling and integration.

This chapter introduces the basic concepts and methods of data profiling and data integration, which lay the foundations for all methods in this thesis. Data profiling is the collection and calculation of statistics and metadata about one or more data sources. It allows data analysts to gain more insights about the data sources and supports the implementation of efficient data processing algorithms. Data integration is the process of merging data from disparate and heterogeneous sources into a single, consolidated dataset. This process consists of schema and data matching methods, which generate correspondences between the elements of the different data sources, schema integration and schema mapping methods, which specify how to merge their schemata, and data fusion methods, which determine how to consolidate conflicting values.

This chapter is organised as follows. Section 2.2 gives a preliminary introduction to the relational data model. Section 2.3 then introduces data profiling for a single source and multiple sources setting. Finally, Section 2.4 describes the data integration process and its individual steps.

2.2 Preliminaries: Relational Data Model

This section introduces the relational data model [Codd, 1970] and defines its terminology and concepts according to Ullman [Ullman, 1989] and Date [Date,

2012]. The relational model is the foundation for the interpretation of relational web tables, which are the focus of this thesis, and defines concepts such as functional dependencies and candidate keys, which will play an important role for the method introduced in Chapter 9.

A relation, relational schema, or just schema, defines a set of attributes and their data types. Data records that use this schema are called tuples and sets of such tuples with the same schema are called a relation instance of the respective relational schema. A relational schema with n attributes is said to be of degree or arity n . Relational schemata with 1, 2, 3 or n attributes are also referred to as unary, binary, ternary or n -ary relations.

Definition 1 (Relational Schema) *A relational schema $R = \{A_1, A_2, \dots, A_n\}$ is a set of ordered pairs (N, dom) , the attributes of R , each consisting of an attribute name N and domain dom , which specifies the set of possible values of the attribute.*

Data that conforms to a relational schema is stored in tuples which contain one value for each attribute in the schema. A set of such tuples is called a relation instance. A relation instance with m tuples is said to be of cardinality m .

Definition 2 (Tuple) *A tuple with relational schema R is a set of ordered pairs (A, v) that assign a value v from the corresponding domain to each attribute A in R .*

Definition 3 (Relation Instance) *A relation instance r with relational schema R is a set of tuples with relational schema R .*

Basic operators for relation instances are projection, join, and union. The projection operator creates a new relation instance with the specified subset of attributes. The join operator merges two or more relation instances into a new relation instance with a relational schema that contains all of their attributes. The union operator merges two or more relation instances into a new relation instance that contains all of their tuples.

Definition 4 (Projection) *Let t be a tuple with relational schema R and X be a subset of R . The projection $t[X]$ of t on the attributes of X is a tuple with relational schema X containing just those (A, v) pairs such that A appears in X . The projection of a relation instance is the set of all projections of its tuples.*

Definition 5 (Natural Join) *Let relation instances r_1, r_2 with relational schemata R_1, R_2 be joinable, i.e., be such that attributes with the same name are of the same type. Then the join $r_1 \bowtie r_2$ is a relation instance with relational schema $S_1 \cup S_2$.*

Definition 6 (Union) *Let relation instances r_1, r_2 be unionable, i.e., have the same relational schema. Then the union $r_1 \cup r_2$ is a relation instance that contains all tuples which are in r_1 or r_2 or both.*

Relational constraints can limit the set of possible instances of a relational schema and are used to express additional information about the real-world object that is modelled in the relational schema. Frequently used relational constraints are keys and functional dependencies. Functional dependencies specify the functional relationship between sets of columns in a relation. They specify the set of attributes that uniquely determines the value of another attribute. Based on functional dependencies, the candidate keys of a relation can be defined as the minimal sets of attributes that uniquely identify any tuple of the relation.

Definition 7 (Functional Dependency) *A functional dependency (FD) with respect to a relational schema R is an expression of the form $R : X \rightarrow Y$ where X is called the determinant and Y is called the dependant and $X, Y \subseteq R$. A relation instance r with schema R satisfies this FD if all its tuples t_1, t_2 are such that whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$, otherwise r violates the FD.*

Definition 8 (Candidate Key) *Given a relational schema R , any set $X \subseteq R$ for which holds $X \rightarrow R$ is a superkey of R . If there exists no $Y \subset X$ such that $Y \rightarrow R$, then X is said to be minimal and a candidate key.*

The concepts defined above are rather abstract, and it may be helpful to clarify how they relate to concrete data representations. In general, every relation instance can be represented as a table. Each attribute of its schema corresponds to a column and each tuple corresponds to a row. As the ordering of attributes and tuples is not important, multiple different tables can represent the same relation instance. It is further possible that a table does not represent a relation instance, i.e., if it does not have a schema or contains duplicate rows. This means that, strictly speaking, not all web tables are relation instances. Where necessary, this can be resolved by generating a schema and removing duplicate rows.

2.3 Data Profiling

This section introduces data profiling and its individual tasks for a single data source and for multiple data sources. Single source profiling is concerned with the description of data from a single data source, such as a file or a relational database. Multiple source profiling deals with the analysis of the overlap of different, heterogeneous data sources. The different data profiling tasks are presented along the taxonomy proposed by Naumann [Naumann, 2014] and for each task, the usual applications are described and references to relevant related work are given.

The term “*Data Profiling*” refers to the task of collecting statistics and meta-data about one or more data sources. It is usually the first step for query optimisation, data cleansing, data integration, and data analysis [Naumann, 2014]. The results of data profiling provide insights about datasets that can be used to find regularities and inconsistencies, such as common patterns in the values of a certain column, or help a data expert to familiarise herself with the data in preparation of a data integration or data mining project.

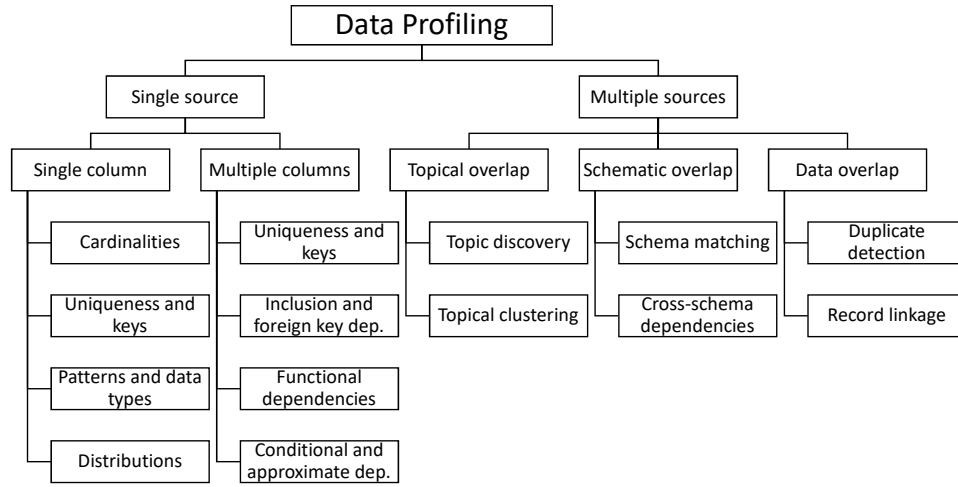


Figure 2.1: Classification of data profiling tasks (reproduced from [Naumann, 2014]).

Figure 2.1 shows a classification of the different data profiling tasks according to Naumann [Naumann, 2014], who primarily distinguishes between single source and multiple source profiling. For single source profiling, tasks are further distinguished between single column metadata, such as cardinalities, data types or value distributions, and multiple column metadata, such as keys and functional or inclusion dependencies. These statistics allow for a basic understanding of the profiled dataset and can further be useful for tasks such as query optimisation, data cleaning or specific data analysis. For multiple source profiling, the sub tasks comprise the analysis of topical, schematic and data overlap. The resulting statistics give insights about the common elements in the data sources, such as common attributes or data records, and are a prerequisite for data integration. In the context of this thesis, the data profiling of multiple sources is of special importance, as its methods are used to create the topical profile of web tables.

2.3.1 Single Source

This section describes the data profiling tasks which are defined for a single source, such as a single dataset or relational database. The tasks are further differentiated between single column profiling and multiple column profiling. While most tasks for single columns are computationally cheap and can be solved with a single scan over the data, the tasks for multiple columns need to consider all possible combinations of columns in a dataset and are hence computationally costly.

The basic statistics for a column can be created by scanning once over the data and reporting different counts such as the total number of values, the number of missing values or the set of unique values and the frequency of each of these values. Such statistics are, for example, used for database and query optimisa-

tion [Mannino et al., 1988, Poosala et al., 1996] or for data cleaning [Rahm and Do, 2000]. A more involved profiling step is to find common value patterns for a column, which can be used to create data cleaning rules and spot anomalous data values [Raman and Hellerstein, 2001].

Statistics for multiple column profiling are computationally more expensive as they need to analyse all possible combinations of columns. Modern algorithms, however, achieve performance improvements by applying pruning to this large search space. The most common analysis is that of functional dependencies [Huh-tala et al., 1999, Yao and Hamilton, 2008] and inclusion dependencies [Bauckmann et al., 2007, De Marchi et al., 2002].

A functional dependency specifies the set of attributes that uniquely determines the value of another attribute. For example, in a relation $\{name, city, phone\}$ the functional dependency $\{phone\} \rightarrow \{name\}$ states the value of the phone number attribute uniquely determines the value of the name attribute. Functional dependencies can be used to determine the candidate keys of a relation, check the data quality, and to perform normalisation of relational schemata into the Second, Third, and Boyce-Codd Normal Form [Codd, 1972].

Inclusion dependencies can be used to find foreign keys between two relations. They specify that the values of a set of attributes in one relation, the foreign key, is a subset of the values of another set of attributes, the key, in a second relation. Knowledge about inclusion dependencies can further be used to avoid update anomalies and improve data integrity.

2.3.2 Multiple Sources

This section describes the data profiling tasks that are defined for multiple sources. Tasks in this area enable users to learn about the common properties and integrability of the data. The difference when dealing with multiple sources compared to single source profiling is the increased degree of heterogeneity. When dealing with data from multiple sources, profiling analyses the degree of this heterogeneity. Heterogeneity can be divided into the three types syntactic, structural, and semantic [Özsu and Valduriez, 2011]:

- **Syntactic Heterogeneity** refers to inconsistent representations of the data, for example through different encoding, different data formats such as CSV and XML, or different value formatting.
- **Structural Heterogeneity** refers to the use of unmatched schemata in the different sources, which means that the same information may be structured differently.
- **Semantic Heterogeneity** refers to the different meaning of the elements in the different data sources. This can show in the form of synonyms and homonyms, differences in the used ontology or ambiguity through imprecise wording.

Discovering syntactic heterogeneity is the focus of traditional data profiling methods, for example by comparing the discovered value patterns for different columns. The discovery of structural and semantic heterogeneities is performed using schema and data matching methods known from the field of data integration. These methods discover correspondences among the schema elements and data records of the different sources, and are discussed in detail in the next section. Profiling methods for multiple sources can use schema matching techniques to determine the *schematic overlap*, i.e., determine to which degree the schemata of the different sources overlap or complement each other. The same holds for the *data overlap* on record-level: profiling methods can use data matching techniques to estimate the total number of described real-world entities in the different sources. Finally, determining the *topical overlap* allows for the exploration of a large pool of unknown sources by matching the sources to a given set of topics.

2.4 Data Integration

This section introduces the data integration process and describes the individual steps in this process. First, the overall process is introduced and the terminology is defined. Then, each step of the process is described and the fundamental methods from the respective areas are introduced.

The goal of data integration is to merge data from multiple sources into a single, complete, and concise dataset [Bleiholder and Naumann, 2009]. Completeness means that all real-world entities that are described in the original sources are also described in the integrated dataset. Conciseness means that no real-world entity is described by more than one record in the integrated dataset. As data integration is not limited to data in a specific data model, such as the relational model, this section uses a more general terminology. The term dataset is used to refer a collection records which have values for different attributes. This corresponds to relation instances, tuples, and attributes in the relational model, but also applies to other data models.

The data integration process, visualised in Figure 2.2, resolves the three types of heterogeneity introduced earlier: syntactic, structural and semantic heterogeneity. The methods that identify the overlapping parts of different datasets are generally referred to as “*Matching*”. Specifically, the identification of schema overlap is referred to as “*Schema Matching*” or “*Schema Alignment*”, and the identification of record overlap is, among many other names, referred to as “*Data Matching*”, “*Record Linking*” or “*Identity Resolution*”. After applying the matching methods, the datasets are merged by transforming the records into a common schema, called “*mediated*” or “*global*” schema and then fused into a single dataset by resolving data conflicts. Data conflicts arise if multiple sources describe the same real-world entity with different values for the same attribute.

The data integration process that is described in this chapter corresponds to a materialised integration, also known as warehousing, where the data from the

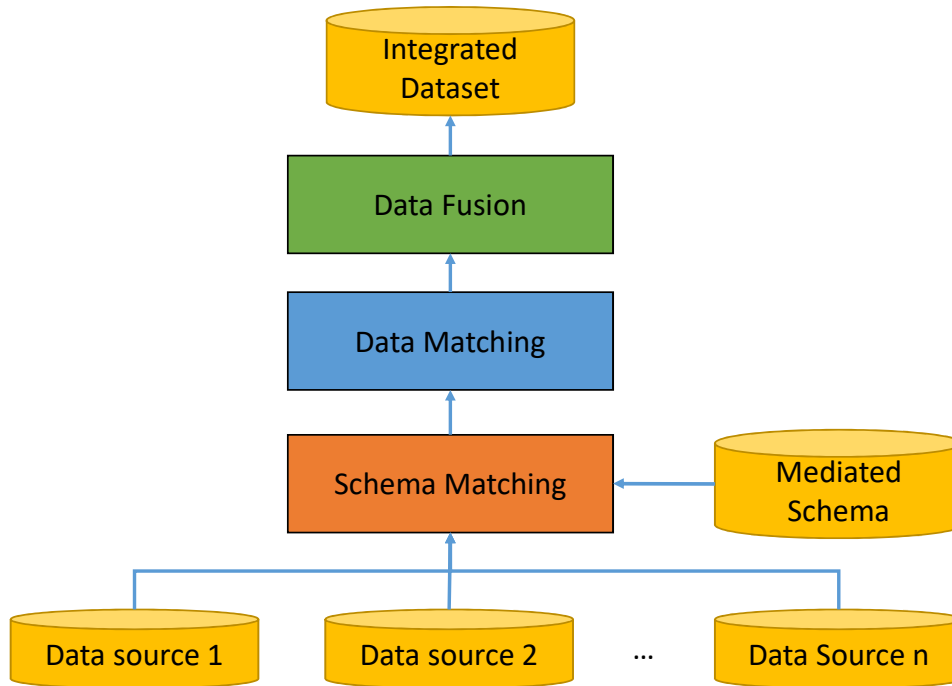


Figure 2.2: The Data Integration Process

disparate sources is loaded into a single system, or warehouse, that can be queried using the mediated schema. The alternative is a virtual integration, where the data resides in the original sources and queries to the mediated schema are translated into queries to the original sources [Doan et al., 2012].

2.4.1 Matching Tasks

This section introduces and defines data and schema matching and discusses the challenges that methods for these tasks are faced with. Matching is the process of aligning the elements (instance or schema level) of one or multiple datasets. Such an alignment is created by the calculation of similarity scores between the elements and by selecting matching elements based on these similarity scores. The result of a matching process is a set of correspondences between semantically equivalent elements. The literature on the topic usually differentiates between data matching on the instance level and schema matching on the schema level.

Data matching is the task of finding correspondences between the records of one or more two datasets that represent the same real-world entity [Newcombe et al., 1959]. It has a large range of applications, from merging duplicate entries in a customer database to comparing the prices of offers from different online shopping web sites. The need for data matching arises from the fact that there is no mechanism to assign globally valid identifiers to entities which are consistently used in all datasets. Rather, each data producer defines how to identify entities in-

dividually, and data matching methods try to infer which records describe the same real-world entities based on the data that is provided. This is a long standing problem and the different research directions are described in more detail in various surveys on the topic [Gu et al., 2003, Winkler, 2006, Elmagarmid et al., 2006].

Definition 9 (Data Matching) *Given two datasets a, b , find the correspondences $C_D \subseteq a \times b$ between all records that represent the same real-world entity, i.e., are semantically equivalent [Fellegi and Sunter, 1969].*

To decide which records represent the same real-world entity, a decision function, as stated in Equation 2.1, needs to be defined. In practice, the decision function is defined over the comparison vector of two records, which encodes the similarity of the records [Fellegi and Sunter, 1969], and results in the probability of the two records representing the same real-world entity.

$$m : r \times s \rightarrow \begin{cases} 1 & \text{semantically equivalent} \\ 0 & \text{not equivalent} \end{cases} \quad (2.1)$$

Analogously to data matching, schema matching is the task of finding correspondences among schema elements, such as attributes or relations. Schema matching is required to discover which attributes or combinations of attributes in different sources are semantically equivalent and can be merged. As for data matching, the necessity for schema matching arises from the fact that different data providers name semantically equivalent attributes differently, use the same name for distinct attributes, or model the same information with a different structure. A more detailed discussion of schema matching methods can be found in the various surveys and text books on the topic [Rahm and Bernstein, 2001, Bernstein et al., 2011, Gal, 2011].

Definition 10 (Schema Matching) *Given two schemata A, B , find the correspondences $C_S \subseteq A \times B$ that map each attribute in A to its semantically equivalent attribute in B [Rahm and Bernstein, 2001].*

The challenge for both matching tasks is to find a measure for *semantically equivalent*, i.e., define the decision function m . Finding the decision function is hard, because the semantics of the datasets are often not fully captured by their schema and records, but can also be implicit [Miller et al., 2000]. The specific challenges for matching techniques are:

- **Language:** The naming of attributes as well as data values of records is often based on natural language to express their meaning. Understanding the semantic relation between such names can be challenging due to the use of synonyms, homonyms, or hypernyms and the ambiguity that they create. Synonyms are different terms that refer to the same concept. A homonym is a single term that is used for different concepts. Hypernyms are terms which are more generic than their corresponding hyponyms [Özsu and Valduriez, 2011].

- **Impreciseness:** The same expression in different sources may refer to different semantic concepts, but this is often not stated explicitly. For example, one source may state the net price of a product and another one states the gross price, but both sources refer to the respective attribute by the same name “*price*” [Özsu and Valduriez, 2011].
- **Modelling:** The same information can be modelled in various ways in different sources. For example, an attribute can be multi-valued in one source, while the same information is represented as multiple Boolean attributes in another source [Busse et al., 1999].
- **Data Quality:** Different sources might contain data errors or are missing attribute values, which results in a low data overlap and makes the comparison of records challenging [Köpcke and Rahm, 2010].

To define possible decision functions, a variety of similarity measures has been proposed, which can be used to define matchers that approximate the correct decision function. An overview of possible similarity measures is given in Section 2.4.2 and Section 2.4.3 introduces the architecture of matchers which use these similarity functions to approximate the decision function.

2.4.2 Similarity Calculation

This section defines the notion of similarity measures and introduces several similarity measures for various data types. These measures are used to calculate the similarity of data values or the names of schema elements and are essential for all matching algorithms.

Semantically equivalent values in different data sources are often not syntactically equivalent, which makes them hard to detect. For example, names might be stated as “*firstname lastname*” or alternatively as “*lastname, firstname*”. These differences can be accounted for by using a similarity measure rather than checking values for exact literal equality. A similarity measure calculates the degree of similarity between two values. Analogously, a distance measure calculates the degree of dissimilarity between two values and is the inverse of a similarity measure.

Definition 11 (Similarity Measure) *A similarity measure is a function that maps pairs of values from a given domain to a real value in the range $[0, 1]$ where larger values indicate greater similarity [Cohen et al., 2003].*

String Similarity. For string values, several measures have been proposed that either tokenise the string first and compare the sets of tokens, or apply a character-based comparison. Surveys of the multitude of different approaches are given by Cohen et al. [Cohen et al., 2003] or Elmagarmid et al. [Elmagarmid et al., 2006]. The Levenshtein Distance [Levenshtein, 1966], as an example for a character-based measure, counts the minimal number of edits needed to transform one string into

the other. Edits can be deletion, insertion, or substitution and the minimal number of edits can be determined using a dynamic programming algorithm [Navarro, 2001]. The Levenshtein Similarity $\text{sim}_{\text{lev}}(a, b)$ between two strings a, b can be obtained by dividing the distance $\text{dist}_{\text{lev}}(a, b)$ by the length of the longest of the two strings:

$$\text{Levenshtein : } \text{sim}_{\text{lev}}(a, b) = 1.0 - \frac{\text{dist}_{\text{lev}}(a, b)}{\max(|a|, |b|)} \quad (2.2)$$

Numeric Similarity. Numeric similarity can be calculated as the absolute or percental difference between two numeric values a and b . In this case, a maximum difference d_{max} must be specified by the user to normalise the similarity value to the range $[0, 1]$:

$$\text{Absolute Difference : } \text{sim}_{\text{abs}}(a, b) = \begin{cases} 1.0 - \left(\frac{|a-b|}{d_{\text{max}}}\right) & \text{if } |a-b| < d_{\text{max}} \\ 0.0 & \text{otherwise} \end{cases} \quad (2.3)$$

An alternative that does not require a user-specified parameter is to calculate the ratio between the absolute values of both numbers [Rinser et al., 2013]:

$$\text{Ratio : } \text{sim}_{\text{ratio}}(a, b) = \frac{\min(|a|, |b|)}{\max(|a|, |b|)} \quad (2.4)$$

Variations of this measure may penalise non-exact matches:

$$\text{Adjusted Ratio : } \text{sim}_{\text{adj.ratio}}(a, b) = \begin{cases} 1 & \text{if } a = b \\ \frac{1}{2} \frac{\min(|a|, |b|)}{\max(|a|, |b|)} & \text{otherwise} \end{cases} \quad (2.5)$$

Date Similarity. The comparison of dates can be performed by using a numeric measure to compare the year-part of each date, as shown in Equation 2.6, or by transforming both dates into the number of days between the date and specific reference date.

$$\text{Year : } \text{sim}_{\text{year}}(a, b) = \text{sim}_{\text{numeric}}(\text{year}(a), \text{year}(b)) \quad (2.6)$$

Alternatively, each part of the date, i.e., day, month, and year, can be compared separately and the final similarity value is a weighted combination of the individual scores:

$$\begin{aligned} \text{Weighted : } \text{sim}_{\text{weighted}}(a, b) = & \alpha \text{sim}(\text{day}(a), \text{day}(b)) + \\ & \beta \text{sim}(\text{month}(a), \text{month}(b)) + \\ & \gamma \text{sim}(\text{year}(a), \text{year}(b)) \end{aligned} \quad (2.7)$$

Set Similarity. Set similarity measures can be applied for multi-valued attributes, for the comparison of all values of two attributes, or for sets of tokens that are created from strings, such as the set of all words or n-grams [Ullmann, 1977, Ukkonen, 1992]. The similarity of two sets A, B can be measured by setting their intersection in relation to the size of the smaller set (overlap similarity) or to the size of their union (Jaccard similarity) [Jaccard, 1912].

$$\text{Overlap : } \text{sim}_{\text{overlap}}(A, B) = \frac{|A \cap B|}{\min(|A|, |B|)} \quad (2.8)$$

$$\text{Jaccard : } \text{sim}_{\text{jaccard}}(A, B) = \frac{|A \cap B|}{|A \cup B|} \quad (2.9)$$

For comparisons of long strings based on word tokens, it can be useful to allow minor differences, such as edit errors, between the tokens. The generalised Jaccard similarity hence uses an inner similarity measure sim' when determining the intersection of the two compared sets: if the inner similarity is above a threshold θ , then the elements are considered equal [On et al., 2007]. Note that if $\text{match}(A, B) = A \cap B$ in Equation 2.10, this is equal to the Jaccard Similarity in Equation 2.9.

$$\text{Generalised Jaccard : } \text{sim}_{\text{gen.jac}}(A, B) = \frac{|\text{match}(A, B)|}{|A| + |B| - |\text{match}(A, B)|} \quad (2.10)$$

with

$$\text{match}(A, B) = \{(a_i, b_j) | a_i \in A \wedge b_j \in B : \text{sim}'(a_i, b_j) \geq \theta\} \quad (2.11)$$

Vector Similarity. Strings can further be compared by first transforming them into vectors. The TF-IDF method for vectorisation of text documents from the field of information retrieval [Manning et al., 2008] transforms a string into a sparse numeric vector that assigns a real value to every token of the string with a fixed vocabulary. In such a vector, the values indicate the importance of each token for the comparison. The similarity between two strings based on TF-IDF can then be calculated as the inner product between their vector representations [Chaudhuri et al., 2003]. As the inner product corresponds to the cosine of the angle between the vector representations, it is also often referred to as “*Cosine Similarity*”.

$$\text{Cosine Similarity : } \text{sim}_{\text{cosine}}(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{|\vec{A}| |\vec{B}|} = |\vec{A}| |\vec{B}| \cos \angle \vec{A} \vec{B} \quad (2.12)$$

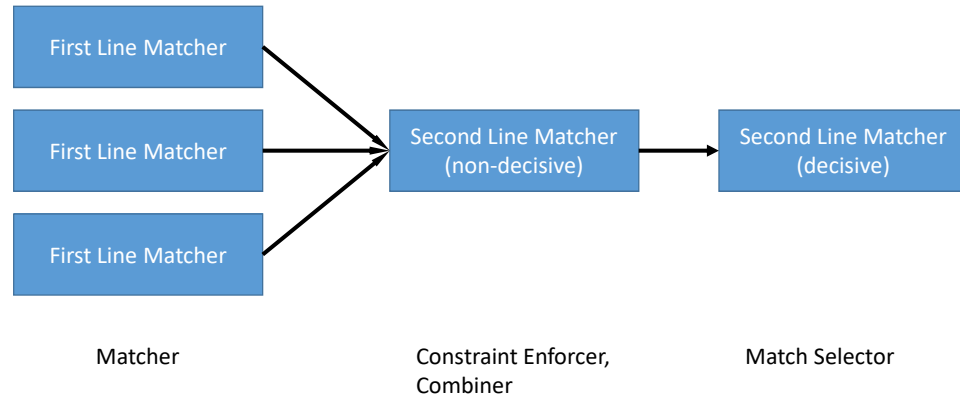


Figure 2.3: Matcher components according to the classification systems of Gal and Sagi, and Lee et al.

2.4.3 Matcher Architecture

This section introduces the terminology and methods that are used to define matchers for schema matching and data matching. First, a classification of matcher components is presented and the general architecture is introduced. Then, the components for the different matching steps are introduced and common methods are described.

A single similarity measure is often not enough to model a high-quality matching system. For example, two cities might have the exact same name but are not the same real-world entity, like “*Paris (France)*” and “*Paris (USA)*”. In this case, a string matcher that compares the city name will find perfect similarity, while another matcher that compares the country will find no similarity. Hence, multiple similarity measures can be combined into complex matchers that improve the quality. Such matchers may consist of multiple steps that are executed sequentially, where each step re-uses the result of the previous step, or in parallel, where each step use the same input, with a subsequent step that combines the different results.

Gal and Sagi [Gal and Sagi, 2010] distinguish between first-line and second-line matchers, as well as between decisive and non-decisionive matchers. First-line matchers receive the original datasets as input and output a similarity matrix, while second-line matchers receive one or more similarity matrices as input. Decisive matchers determine matches between the elements in the input data, i.e., make a decision, while non-decisionive matchers only calculate a similarity among these elements. Figure 2.3 shows an example matcher architecture according to this classification as well as the classification proposed by Lee et al. [Lee et al., 2007].

First Line Matcher. A first line matcher compares pairs of records or attributes and calculates a similarity score for each pair. Such matchers may apply a simple similarity function or evaluate a complex matching rule. If a similarity threshold is enforced by the matcher, it is a decisive first-line matcher.

Second Line Matcher. A second line matcher has a decision function that considers the similarity calculated by one or more first line matchers or other second line matchers as input. Such a matcher can, for example, prune the similarity matrix below a threshold or apply a global matching by calculating the maximum-weight bipartite matching [Galil, 1986] based on the similarity values.

Alternative Classifications. Lee et al. [Lee et al., 2007] classify matchers into the categories “*matcher*”, “*combiner*”, “*constraint enforcer*”, and “*match selector*”. A matcher corresponds to a first-line matcher, while the remaining categories describe different types of second-line matchers. A combiner merges the similarity matrices created by multiple other matchers, for example by averaging. A constraint enforcer might apply domain-specific constraints that remove improbable correspondences and a match selector is a decisive second-line matcher.

Reducing the Number of Comparisons

Matchers look for semantically equivalent elements in the cross product of their inputs, i.e., all possible combinations of records or attributes. This input space grows quadratically with the size of the datasets and quickly becomes computationally infeasible, especially for the data matching task. This section introduces methods to reduce the number of candidate pairs of records, which are executed before the actual matcher and are computationally cheaper.

Blocking. To find approximately matching pairs, one or multiple highly discriminative values of the records are used to define a blocking key attribute and all records with the same value for this attribute form a block [Elmagarmid et al., 2006]. The matcher is then only applied to all pairs that can be generated from the same block, and records from different blocks are assumed to be non-matches. For example, a blocking key attribute might contain the first three letters of a customer’s last name and the first two digits of her zip code. A drawback of this method is that correct matches are missed if the records are placed in different blocks, for example if an unsuitable blocking key is chosen. To prevent this, multiple runs with different blockings keys can be executed.

Sorted-Neighbourhood Method. A method to increase the completeness of a blocking step is the Sorted-Neighbourhood Method [Hernández and Stolfo, 1998]. Here, all blocking key values are sorted and then a sliding window is applied to the sorted records. This allows for the comparison of records with similar, but not exactly matching blocking key values. It has been shown by Draisbach and Naumann [Draisbach and Naumann, 2009] that it is possible to generalise both approaches to a “*Sorted Blocks*” method, using a parameter that in its extreme settings yields the two different approaches.

Inverted Index. For fast comparisons, an efficient technique is the use of an inverted index [Cohen, 2000]. In such an index, the record values or substrings of these values, such as n-grams, are listed with references to all records that contain this value. A join over these indexed terms then results in the sets of all records that contain common values. It has further been shown how these indices can be pruned to speed up the join computation [Carmel et al., 2001, Sarawagi and Kirpal, 2004]

Matching Approaches

This section introduces different types of matchers. Each of these types takes a different approach to combining the similarity scores that are calculated over pairs of records or attributes and form the comparison vector.

Matching Rules. To determine the similarity of records, matching rules define how to combine the similarities of the individual record values. Hand-written matching rules usually take the form of a “*linear combination*” or “*rule set*”. A linear combination weights the similarity of n different attributes A_i for two records r_1, r_2 and aggregates them into a weighted sum, as shown in Equation 2.13 [Doan et al., 2012].

$$\text{sim}_{\text{record}}(r_1, r_2) = \sum_{i=1}^n \theta_i \cdot \text{sim}_i(r_1[A_i], r_2[A_i]) \quad (2.13)$$

A rule set contains rules that specify conditions for a match and allow for a non-linear combination of the attribute similarities, as shown in Equation 2.14.

$$\text{sim}_1(r_1[A_1], r_2[A_1]) > \theta_1 \wedge \text{sim}_2(r_1[A_2], r_2[A_2]) > \theta_2 \Rightarrow \text{match} \quad (2.14)$$

Matching rules can be designed manually by experts [Wang and Madnick, 1989], which usually results in high accuracy, but also requires a high manual effort. If training data is available, matching rules can also be learned using machine learning methods [Bilenko et al., 2003, Cochinwala et al., 2001, Isele and Bizer, 2012]. For example, a linear combination can be represented as a logistic regression model and a rule set can be represented as a decision tree model. To reduce the requirements to the training dataset, active learning methods have also been used to generate matching rules [Isele and Bizer, 2013].

Duplicate-based Matching. In schema matching, the similarity of attributes can also be determined by comparing the values of duplicate records, i.e., by re-using the results of a preceding data matching step [Bilke and Naumann, 2005, Zhou et al., 2007]. For each record correspondence, the similarity of the values for all attribute combinations is calculated. These similarity values are then averaged over all record correspondences C_D to determine the similarity score for the respective

attribute pair, as shown in Equation 2.15. This usually gives better results than comparing the set of all values of an attribute, because the similarity is only calculated for attribute values that describe the same real-world entity.

$$\text{sim}_{\text{duplicate}}(A_1, A_2) = \frac{1}{|C_D|} \sum_{(r_1, r_2) \in C_D} \text{sim}(r_1[A_1], r_2[A_2]) \quad (2.15)$$

Global Matching

The matchers presented above compare all record pairs individually, and may hence produce multiple correspondences for a given record. It is, however, often desirable to limit the result to the most confident correspondences or even enforce a 1:1 mapping, where each record can only correspond to a single other record [Doan et al., 2012]. This section introduces approaches for such a global matching, which are decisive second-line matchers.

Top-K Matches. A computationally cheap approach is to limit the set of correspondences for each record or attribute to the k correspondences with highest similarity score. This does, however, not enforce a 1:1 mapping as multiple records from one dataset might have their correspondence with the highest similarity score to the same record in the second dataset.

Maximum-Weight Matching. The matching problem between two datasets can be modelled as a bipartite graph, in which the elements, records or attributes, of each dataset are vertices and weighted edges connect the vertices of different datasets. The weight of each edge is the similarity as calculated by a matcher. In such a graph, the maximum-weight matching is the subset of the edges with the largest sum of weights, such that each vertex is only connected to a single other vertex [Galil, 1986].

2.4.4 Data Fusion

After schema and data matching methods have established which attributes and records in different datasets correspond to each other, data fusion methods are used to create an integrated dataset. This section introduces the task of data fusion and discusses how it can be implemented using relational operators. Further, different strategies for the handling of conflicting data values are presented.

Once two or more datasets have been matched, and their schemata have been mapped to each other, the data can be combined into an integrated dataset. Combining here means to merge the attributes and records from the different sources into a single dataset. When merging records based correspondences from a data matching step, different sources might provide conflicting values. Such data conflicts can either be escalated to an end user or be resolved using various strategies.

Relational Operators

Datasets can be merged through two different types of relational operators: by joining them or by creating their union. In the context of data integration, these operations have to deal with a higher level of heterogeneity than in a single-database scenario, i.e., corresponding attributes can have different names and corresponding records might not share a common key. This section introduces variants of the relational join and union operators and discusses their implication for the data fusion process. A more detailed discussion of these operators is given by Bleiholder and Naumann [Bleiholder and Naumann, 2009].

Join. Both the equi-join on key attributes $\bowtie_{Key=Key}$ and the natural join \bowtie combine the records of two datasets on matching values. They are uniqueness preserving, meaning that if the values for the attributes in the join condition are unique in the sources, they are also unique in the joined result. But, as they only produce a record if values from both datasets can be matched, records without counterpart in one of the datasets are missing and these joins are hence not object preserving and affect the completeness of the integrated dataset. The full outer join on key attributes $|\bowtie|_{Key=Key}$ is, in addition to being uniqueness preserving, also object preserving, as records without counterpart are included in the result by padding them with null values. However, if more than two relations are integrated, the result of multiple outer joins depends on their order, i.e., the outer join operator is not associative. Further, matching attributes are only merged if they have the same name, in all other cases, the join result contains redundant attributes. If the joined sources provide exactly the same values, this can be resolved by renaming the attributes before the join. If, however, the sources provide conflicting information, additional functions must be used to handle the conflicts as described in the next section.

Union. The union operator \cup is only defined for datasets with the same schema: all records of both datasets are added to the result and exact duplicates are removed. For different schemata, the outer union operator \oplus first adds missing attributes and pads them with null values to align the schemata and then applies the union operation. As this can result in records which only differ with respect to null values, another useful variation is the minimum union \oplus , which extends the outer union and also removes subsumed records, i.e., records that always have the same value as another record or null [Galindo-Legaria, 1994]. The union operators are object preserving, as all original objects are present in the result. They are not uniqueness preserving, as exact duplicates, but no conflicting tuples, are removed from the result. This can be avoided and conciseness of union results can be improved with the use of a grouping operation on the object identifier (possibly obtained through data matching) and the conflict handling methods discussed in the following section.

Conflict Handling

After merging the datasets with one of the operators described above, data conflicts can occur if different datasets provide contradicting values. This section introduces strategies for handling such conflicts. The discussed strategies are classified according to Bleiholder and Naumann [Bleiholder and Naumann, 2009], who distinguish between three different types of conflict handling strategies: “*Conflict Ignorance*”, “*Conflict Avoidance*”, and “*Conflict Resolution*”.

Conflict Ignorance. Data fusion with conflict ignorance does not make any decision in case of a data conflict and might not even be aware of such conflicts. The conflict must then be resolved by an application or by the user. This is, for example, the case if datasets are merged by a union operation and can result in redundant records which describe the same entity in the integrated dataset.

Conflict Avoidance. Conflict avoidance functions decide how to handle inconsistencies before seeing the actual values, and hence must not necessarily be aware of conflicts. These functions are specified for each attribute in the integrated dataset in advance and applied to all conflicting records. Examples are the `coalesce` function, which returns the first non-null value or a function that always chooses the value from the preferred data source.

Conflict Resolution. Conflict resolution functions consider the actual values and metadata before deciding how to resolve a data conflict. They can be further divided into deciding and mediating functions. Deciding functions choose one of the existing values, for example by a majority vote. Mediating functions can choose a value that does not necessarily exist in the original sources, for example by taking the average of all values.

The final step of the data integration process, data fusion, uses the original datasets and the results of schema and data matching steps as its input and produces a single, integrated dataset. The schema correspondences obtained through schema matching are used to align the attributes of the datasets, i.e., to rename their attributes before merging them using a join or union operation. Similarly, the record correspondences obtained through data matching are used to insert common key values which allows the corresponding records to be joined or grouped together. After the source datasets have been merged and data conflicts have been resolved, the data integration process is completed and the result is a single, integrated dataset.

2.5 Conclusion

This chapter introduced the topics data profiling and data integration. Data profiling is concerned with the statistical analysis of data sources for the purposes of query optimisation, data cleansing, data integration, or general data analysis. The different statistics can be categorised based on the number of sources that are analysed. For single source profiling, statistics of attributes and data values are of interest and can be used to optimise data access or improve the data quality. For multiple source profiling, the overlap with respect to topics, schemata, and data records is of interest and can give insights about the integrability of the data sources. Methods for multiple source profiling are hence those from the area of data integration.

Data integration considers the problems of finding corresponding elements among the schemata and records of disparate data sources as well as their fusion into a single, integrated dataset. The methods used find such correspondences are generally referred to as matching methods, which employ a variety of different similarity measures and architectures of matching components. Using the discovered correspondences, the datasets can then be merged by data fusion methods, which combine the datasets and resolve data conflicts.

These methods are the foundation of all methods that are presented in this thesis. Chapters 5, 7, 8, and 9 will present methods for data and schema matching that find correspondences between web tables and a knowledge base as well as correspondences among web tables. Chapters 4, 6, 8, and 9 will further present data profiles that give a detailed understanding of the contents of the web table corpora introduced in Chapter 4 and show how these web tables can be used for knowledge base augmentation.

Chapter 3

Knowledge Bases

3.1 Introduction

Knowledge bases store structured data about real-world entities and their properties and are nowadays employed in many use cases. The most prominent one might be web search, where knowledge bases are used to detect the mentions of entity and property names in the user's keyword query to try and understand the user's intent rather than just matching the keywords to web pages. Today's web search engines can already match a large number of different queries to their knowledge bases and even provide exact answers, for example about the birth date of popular people, rather than showing a list of web pages that likely contain this answer. This is only possible if the used knowledge base has a large coverage of entities and their properties, and there is hence a significant amount of research concerned with the construction and augmentation of such knowledge bases.

This chapter first gives an overview of the RDF data model, which is used by many knowledge bases to store their data. Then, the most commonly used knowledge bases in research, i.e., DBpedia, YAGO, Wikidata, etc., are introduced and compared. Such knowledge bases are often based on an automatic extraction of statements from Wikipedia, but differ in the way the knowledge is organised and how additional knowledge can be added to the knowledge base. While DBpedia and YAGO are constructed through automatic extraction only, Freebase and Wikidata rely on user contributions to accumulate additional knowledge.

For DBpedia, which is used in all following chapters, such user contribution is used to create a high quality mapping from the Wikipedia pages to a hand-crafted ontology, which is used during the automated extraction of the knowledge base. The extracted knowledge base contains one entity for each page in Wikipedia and has a strong focus on describing people of public interest. It is, however, far from being complete and knowledge base augmentation methods are needed to gather additional knowledge. These methods might either fill-in missing values, add additional entities, or find new properties for the existing entities in the knowledge base.

This chapter is organised as follows. Section 3.2 provides a preliminary introduction of the RDF data model. Then, Section 3.3 introduces knowledge bases, gives an overview of the most common knowledge bases, and describes DBpedia, the knowledge base that is used throughout this thesis, in more detail. Finally, Section 3.4 introduces the topic of knowledge base augmentation, describes its different tasks, and discusses approaches from the literature which are not based on web tables.

3.2 Preliminaries: RDF Data Model

This section introduces the RDF data model and is based on the *W3C RDF 1.1 Primer* and the *RDF 1.1 Concepts and Abstract Syntax Recommendation* [Manola et al., 2004, (W3C), 2014]. It is the data model that is used by many knowledge bases such as DBpedia, which is used throughout this thesis. The terminology used to define the RDF data model differs significantly from that of the relational model introduced in Section 2.2. Throughout the thesis, the terms of the corresponding data model are used when referring to the relational or the RDF data model, respectively.

The RDF data model is a graph-based data model and specifies how to make statements about resources using a simple format, called a triple:

`<subject> <predicate> <object>.`

These triples describe directed arcs between the subject and object, labelled with the predicate. Subject, predicate as well as object can be represented by Internationalized Resource Identifiers (IRIs), which uniquely identify a resource. For example, the statement that the population of Germany is 82 million can be expressed by the following RDF triple:

`dbr:Germany dbo:populationTotal 82175700^^xsd:integer.`

The preceding triple uses the namespace aliases `dbr`¹, `dbo`², and `xsd`³ for brevity. Its subject and predicate are identified by IRIs, but its object is a so-called literal, which consists of a literal value, here “82175700”, and a data type, “`xsd:integer`” in the example, that specifies how to interpret the literal value.

Definition 12 (Triple) *An RDF triple consists of three components: the subject, which is an IRI or blank node; the predicate, which is an IRI; the object, which is an IRI, a literal or a blank node.*

¹<http://dbpedia.org/resource/>

²<http://dbpedia.org/ontology/>

³<http://www.w3.org/2001/XMLSchema>

Definition 13 (Literal) A literal consists of two or three elements: a lexical form, being a Unicode string; a data type IRI, being an IRI identifying a data type that determines how the lexical form maps to a literal value; if and only if the data type IRI is `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`, a non-empty language tag.

Definition 14 (Blank Node) Blank nodes are disjoint from IRIs and literals. Otherwise, the set of possible blank nodes is arbitrary.

A set of RDF triples forms an RDF graph. As the same IRI can appear in multiple triples in any of the subject, predicate, and object position, it is possible to create connections between triples. A simple RDF graph is shown in Figure 3.1.

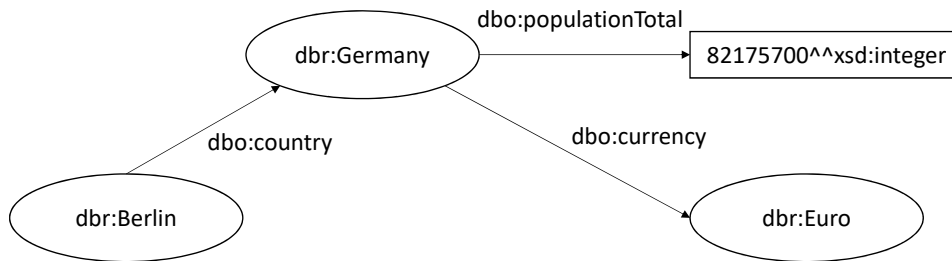


Figure 3.1: An example of an RDF graph.

Definition 15 (RDF Graph) An RDF graph is a set of RDF triples. The set of nodes of an RDF graph is the set of subjects and objects of triples in the graph. It is possible for a predicate IRI to also occur as a node in the same graph.

The RDF data model is used for many of the knowledge bases that will be introduced in section 3.3. These knowledge bases define their own vocabulary to describe the stored data based on the modelling constructs that are available through RDF and RDF Schema [Brickley et al., 2014]. Some of these constructs are the following predefined classes and properties, where the namespace aliases `rdf`⁴ and `rdfs`⁵ are used for brevity. A class in RDF is a group of resources and also is a resource by itself, i.e., a class can be referred to in the subject or object position of a triple. A property in RDF defines a relation between the subject resources and object resources.

- `rdfs:Class`: A resource that identifies a class in RDF. It is the class of all resources that are RDF classes.
- `rdfs:Property`: A resource that identifies a property in RDF. It is the class of all resources that are PDF properties.

⁴<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

⁵<http://www.w3.org/2000/01/rdf-schema#>

- `rdf:type`: A property that states that the subject is an instance of the class identified by the object.
- `rdfs:subClassOf`: A property that states that the subject is a subclass of the object.
- `rdfs:subPropertyOf`: A property that states that the subject is a sub-property of the object.
- `rdfs:domain`: A property that specifies that the domain of the property identified by the subject is an instance of the class identified by the object.
- `rdfs:range`: A property that specifies that the range of the property identified by the subject is an instance of the class identified by the object.
- `rdfs:label`: A property that may be used to provide a human-readable version of a resource's name.

Classes in RDF define a set of resources with some common semantics. They roughly correspond to relations in the relational model as the same set of properties can be used for all resources that are an instance of the same class, just as all tuples for the same relation use the same set of attributes. Properties in RDF, when used in the predicate position of triples, correspond to attributes in the relational model. It is important to note that in RDF terminology, the domain of a property is the type of the subjects of triples using this property. In the relational model, however, the domain of an attribute specifies the set of possible values, which corresponds to the range of a property in RDF.

3.3 Overview of Common Knowledge Bases

This section gives an introduction to knowledge bases by describing the most commonly used knowledge bases and the sources from which they are created.

The improvements made in automatic fact extraction and the growths of collaborative information sharing resources such as Wikipedia have led to the development of several large-scale knowledge bases [Weikum and Theobald, 2010]. These are large databases of structured knowledge, which are created through the extraction and integration of data from web sources. The goal of such general-purpose knowledge bases is to collect the human knowledge and make it accessible to both human users and machines, allowing them to formulate complex queries and perform reasoning over the data.

Recently, the term knowledge graph has been used more frequently instead of the term knowledge base, but there is no clear definition or consensus among researchers about the meaning of these terms. Ehrlinger and Wöß compare different uses of the terms and define them as follows: “A *knowledge graph* is a *knowledge-based system that contains a knowledge base and a reasoning engine*” [Ehrlinger and Wöß, 2016]. According to their terminology, a knowledge base is an ontology, which is a machine-readable definition of an abstract model of (a part of) the

real world that defines the relevant concepts as well as constraints on the use of these concepts. They further identify the “*collection, extraction, and integration of information from external sources*” as an essential characteristic of a knowledge graph.

For the purpose of this thesis, the term knowledge base will be used to refer to a machine-readable representation of entities, their properties and a taxonomy of classes in which they are organised. Further, only large-scale, general-purpose, cross-domain knowledge bases are discussed in the following, as only these knowledge bases are useful for a general profiling of the contents of web table corpora. Table 3.1 compares the knowledge bases that are introduced in the following paragraphs with respect to number of entities, triples, classes, and properties. The comparison shows the different characteristics of the knowledge bases. While DBpedia uses a small type hierarchy with rather generic classes, YAGO provides about 500 times more classes for a similar amount of entities and is much more fine-grained, but uses far less different properties. Freebase and Wikidata cover a larger set of entities, as they rely not only on Wikipedia, but also on user contributions to create additional data. Finally, the Google Knowledge Graph is likely the largest knowledge base at the time of writing, but it is not publicly accessible and only little information is available.

Table 3.1: Overview of common knowledge bases (reproduced from [Paulheim, 2017]).

Name	Entities	Triples	Classes	Properties
DBpedia (English)	4 806 150	176 043 129	735	2 813
YAGO	4 595 906	25 946 870	488 469	77
Freebase	49 947 845	3 041 722 635	26 507	37 781
Wikidata	15 602 060	65 993 797	23 157	1 673
Google Knowledge Graph	570 000 000	18 000 000 000	1 500	35 000

DBpedia. The DBpedia knowledge base is extracted from Wikipedia pages through different extractors that convert the page content into RDF [Lehmann et al., 2015]. This enables querying and automated processing of the vast amounts of data that are available in Wikipedia. By interlinking DBpedia with other datasets in the semantic web, DBpedia has become the central dataset in the linked open data (LOD) cloud [Schmachtenberg et al., 2014].

YAGO. Similar to DBpedia, the YAGO knowledge base is also extracted from Wikipedia. However, the approach taken by YAGO differs in its type system and focus on accuracy [Suchanek et al., 2007]. The type hierarchy used by YAGO is based on WordNet and contains almost half a million classes, while the DBpedia

ontology is hand-crafted and contains only 735 classes. Concerning the size of 5 million triples, YAGO is considerable smaller than DBpedia, but it is supposed to have an accuracy above 95%.

Freebase. The Freebase knowledge base puts more emphasis on collaborative collection of knowledge and enables users to add or modify the information in the knowledge base via APIs and a Web interface [Bollacker et al., 2007]. It is seeded from Wikipedia and several other datasets, but the focus is on cumulating more information through the efforts of its users. The project has, however, been shut down in 2014 after it was acquired by Google and used to seed the non-public Google Knowledge Graph.

Wikidata. The Wikidata knowledge base follows the same idea as Freebase, and allows users to collaboratively edit the data of the knowledge base [Vrandečić and Krötzsch, 2014]. Furthermore, Wikidata is used as a data back-end for Wikipedia and an increasing amount of content on Wikipedia pages, such as links between different language versions of the same article, is generated from Wikidata. Beside simple property-value pairs, Wikidata also supports complex statements which cannot be expressed as simple triples. An example is the statement that the *population* of Rome was 2 761 477 as of 2010 based on an *estimation* published by *Istat*. Such complex statements are achieved through the definition of an extensible set of “*quantifiers*” which can be used to add more information to a base triple that states, in this case, the population of Rome.

Google Knowledge Graph. Introduced in 2012, the Google Knowledge Graph is used to enhance the Google search engine [Singhal, 2012]. There, it is used to provide factual answers for certain types of queries and show summaries on the search results page for recognised entities in the user’s query. It is supposedly the largest knowledge base at the time of writing, but not publicly accessible.⁶

This overview over commonly used knowledge bases shows the different approaches that are taken towards organising the human knowledge in a machine-understandable way. While all of the presented knowledge bases are seeded or constructed completely from Wikipedia, they still differ great in the way the data is modelled. This shows, for example, in the size of the used class hierarchy or number of existing properties. The next section provides further details about the DBpedia knowledge base, which is used in the experiments and analyses that are presented in this thesis.

⁶https://search.googleblog.com/2012/12/get-smarter-answers-from-knowledge_4.html

The DBpedia Knowledge Base

This section provides additional details about DBpedia, which is the knowledge base that is used in all experiments in this thesis. For its use throughout this thesis, the relevant parts of the extraction process are explained and the suitability of the knowledge base for the conducted analyses is discussed. Finally, statistics about the contents of DBpedia, i.e., its entities, classes, and properties, are presented.

DBpedia is created by several extractors that extract statements from different parts of Wikipedia pages into RDF. The most relevant extractor for this thesis is the “*Mapping-Based Infobox Extractor*”, which extracts statements from the Infobox of a page using a manually defined mapping into the DBpedia ontology namespace.⁷ This mapping is crowd-sourced and can be edited by users using a separate Wiki, the DBpedia Mappings Wiki.⁸ The manual creation of the mapping results in a higher level of quality for the triples in the DBpedia ontology namespace than for triples of other extractors. For this reason, this thesis exclusively considers the properties defined in this namespace. The second relevant namespace is the resource namespace, which contains the URIs of all entities.⁹ For these entities, there is a one-to-one mapping between each Wikipedia page and an entity.

For the purpose of this thesis, which focuses on a broad profiling of web tables across various topics, the used target knowledge base should cover many topics and be representative for entities of public interest. These two criteria are fulfilled by all knowledge bases that are extracted from Wikipedia, as the articles in Wikipedia cover a very broad range of topics and its guidelines require the subjects of articles to be notable.¹⁰ While this means that any of the knowledge bases described in the previous section are suitable for the analyses in this thesis, DBpedia in particular has been recognised as a common point of reference in the Web of Data with more than 200 datasets referring to DBpedia [Schmachtenberg et al., 2014].

The version DBpedia that is used throughout this thesis, DBpedia 2014, contains a total of 4.5 million entities, 730 classes, and 2795 properties. Figure 3.2 shows the frequency distribution of entities over the largest high-level classes in the DBpedia type hierarchy. The largest class is *Agent*, which contains 44% of all entities, followed by *Place* with 17%, *TimePeriod* with 16%, and *Work* with 9%. This distribution shows that DBpedia has a strong focus on persons of public interest. But, this also means that large parts of the type hierarchy are rather specific and only cover a small subset of the entities in DBpedia. In total, the highest level in the class hierarchy contains 23 classes, but the five largest classes *Agent*, *Place*, *Species*, *TimePeriod* and *Work* cover 92% of all entities.

Figure 3.3 shows the density of several properties for the largest classes in DBpedia, i.e., the percentage of entities that have a value for these properties. The most frequently filled values are those of the *Eukaryote* class, which specify

⁷<http://dbpedia.org/ontology/>

⁸<http://mappings.dbpedia.org/>

⁹<http://dbpedia.org/resource/>

¹⁰<https://en.wikipedia.org/wiki/Wikipedia:Notability>

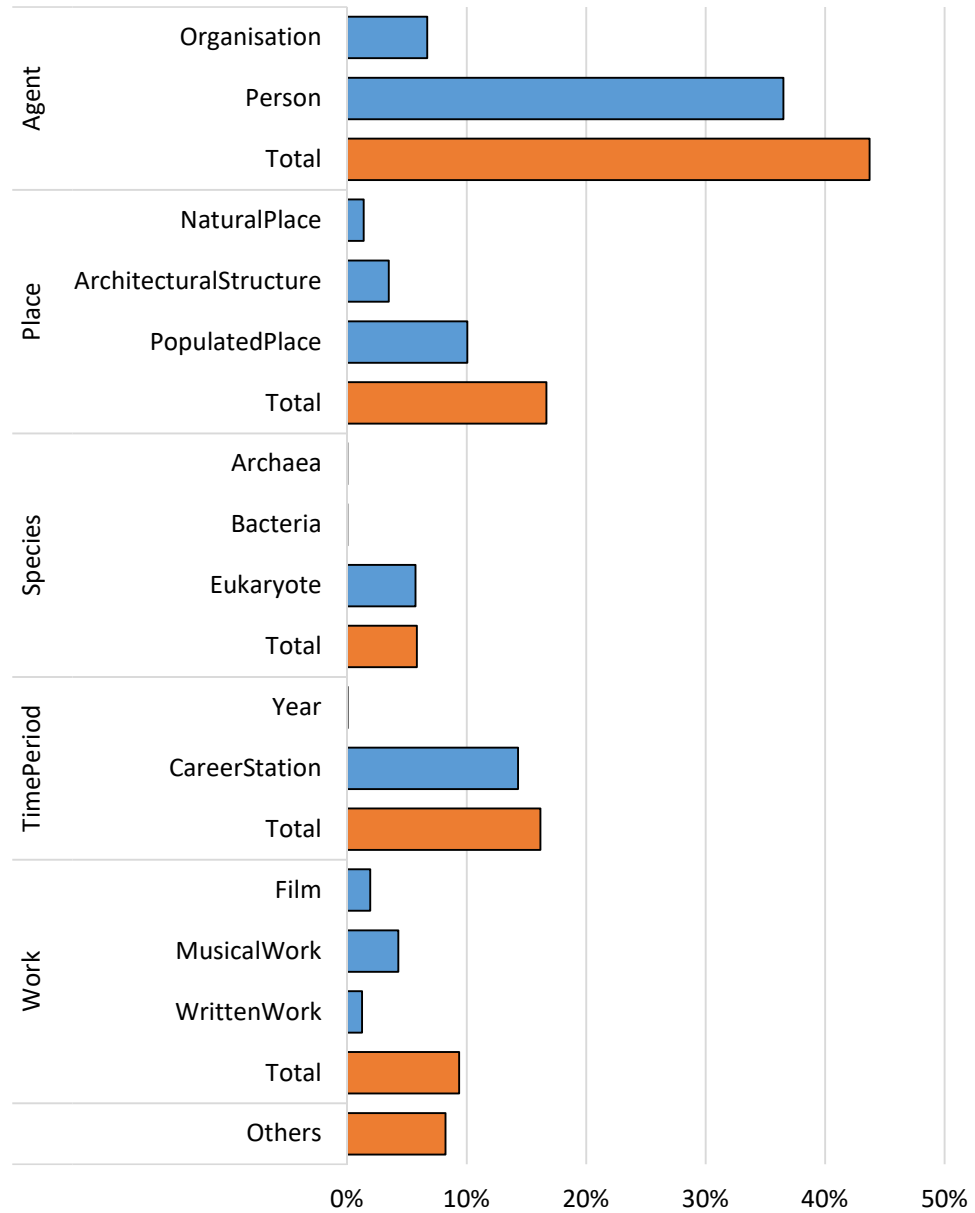


Figure 3.2: Distribution of DBpedia entities by class.

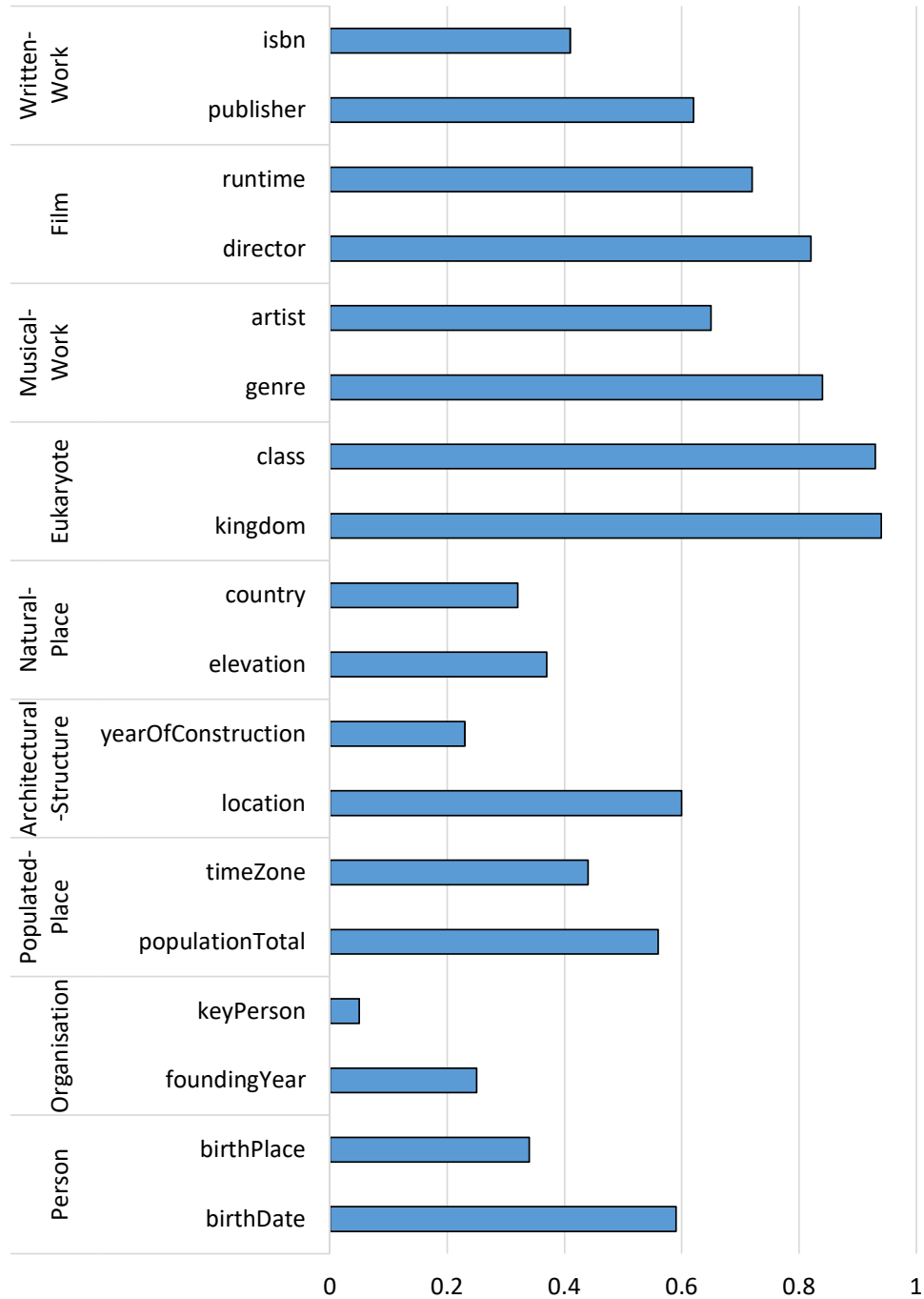


Figure 3.3: Density of selected DBpedia properties.

the classification of the species. Although the figure shows some of the most frequently used properties of the selected classes, an overall rather low density can be observed. For example, only 59% of the persons in DBpedia have a value for their birth date and only 56% of all populated places have a value for population. This shows that there is a clear need for adding missing values to the knowledge base.

3.4 Knowledge Base Augmentation

This section introduces the topic of knowledge base augmentation and discusses approaches from the literature in the areas of Open Information Extraction and Wrapper Induction. An extended discussion of literature for knowledge base augmentation with respect to web tables is given in the related work sections of Chapter 5, 6, and 8.

Knowledge Base Augmentation Tasks

This section introduces the three knowledge base augmentation tasks slot filling, schema extension, and entity-set completion. The overarching goal of these tasks and knowledge base augmentation in general is to extract schematic and factual data from large-scale web sources and add it to existing knowledge bases. The slot-filling task seeks to add values for known properties of existing entities. The schema-extension task has the goal of finding new properties, and the entity-set-completion task aims to find additional entities for the knowledge base.

Slot Filling. The goal of the slot filling task [Surdeanu and Ji, 2014] is to fill-in missing values for known entities and properties. This means either finding a literal value for a given entity and property, such the population of Germany, or finding that a known relation, represented by a property, holds between two known entities, such as the relation “*is capital of*”, which holds between Berlin and Germany.

This task is approached by many information extraction systems, which extract values and relations either from natural language text [Ji and Grishman, 2011] or by learning wrappers for the structure of web pages [Arasu and Garcia-Molina, 2003]. To solve this task, a system must perform linking and disambiguation for entities as well as for properties in multiple data sources and then decide which source to trust in order to produce a final value [Dong et al., 2014a]. An exhaustive overview of such systems is given by Weikum and Theobald [Weikum and Theobald, 2010] as well as Paulheim [Paulheim, 2017].

Schema Extension. The goal of the schema extension task is to find additional properties which can be added to a knowledge base. There are different approaches to the schema extension task, which not only differ in their methodology, but also in their goal and in their evaluation setting.

Open information extraction systems [Fader et al., 2011, Yates et al., 2007] try to discover relations that hold between recognised entities in natural language text. The focus of these methods is on correctly interpreting the natural language sentences and extracting the tokens that represent the relation as new properties. For evaluation, human annotators define which relations should be extracted from a given set of sentences.

A different approach is attribute discovery, which focusses on finding names of new attributes that are relevant for known classes and their entities, and may be added as new properties to a knowledge base [Gupta et al., 2014, Cafarella et al., 2008a]. These methods make no attempt to extract values for the discovered attributes. Instead, attributes are ranked by their relevance for the given classes or entities that should be extended and evaluated with respect to the amount of relevant attributes at a given rank.

A third approach is to find new attributes based on user-provided keyword queries [Yakout et al., 2012, Lehmborg et al., 2015]. Methods in this area search a corpus of heterogeneous data sources, for example consisting of web tables, for attributes that match the query and join them to a dataset that is provided by the user. The evaluation of these methods measures if the correct attributes values were joined to the dataset.

Entity-Set Expansion. The goal the entity-set expansion task is to find missing entities for known classes. Approaches in this area make use of a small set of seed entities and try to find more entities that belong to the same semantic concept in web sources [Ghahramani and Heller, 2006].

This is either done by training extractors for the known entities, for example from structured web sources as in wrapper induction, or by learning textual patterns that indicate class membership, as in “*Countries such as Germany and France*” [Wang and Cohen, 2008, Carlson et al., 2010]. The challenges for these approaches are to create a complete and coherent set of entities. Completeness is challenging because many web sources might only state popular entities, while less popular entities, so-called “*long tail entities*”, are much harder to find [Wang et al., 2015a]. Coherence of the result set means that only entities that correspond to the same semantic concept should be included, i.e., methods have to avoid a “*concept drift*”, which occurs if web sources list similar, overlapping sets of entities that, however, correspond to a different concept.

Approaches from Open Information Extraction

The area of open information extraction from the field of natural language processing is concerned with extracting data from unstructured text. The focus is on detecting entity mentions and relations between entities in natural language sentences. The name “*Information Extraction*” refers to methods that learn an extractor for each relation in a known vocabulary given training examples, while the name “*Open Information Extraction*” is used for methods that identify relations

in phrases without a predefined vocabulary [Yates et al., 2007]. In the following, an overview of approaches from open Information extraction for knowledge base augmentation is given.

NELL. The idea of the NELL (*“Never-Ending Language Learner”*) [Carlson et al., 2010] system is to create a computer agent that runs continuously, reading text from the Web, while collecting a growing knowledge base and improving its ability to understand natural language text. The system has several sub-components that extract fact candidates for known relations from free text and retrieve the names of additional entities from other sources such as HTML list and tables. A *“Knowledge Integrator”* component then decides which of the extracted statements to trust and to integrate with the knowledge base. The system was seeded with an initial knowledge base and collected 242 thousand new statements after running for 67 days. Most of these statements (95%) were category assertions, only 5% were assertions about relations.

PRISMATIC. One of the main limitations of natural language processing for information extraction is the lack of a complete knowledge base that identifies possible relations. Fan et al. hence propose to generate such a resource of relations from a large text corpus. Their PRISMATIC system [Fan et al., 2010] uses natural language processing methods such as dependency parsing, named-entity recognition and co-reference resolution to organise the information contained in unstructured text into so-called *“frames”*. A frame is defined by the words in a sentence and uses a small set of possible relations, called slots, between the words, such as *“subject”*, *“object”*, or *“modifier”*. These frames annotate the components of a sentence with their grammatical function and a set of basic type assertions. The authors argue that, based on the frequency of sub-sets of the possible relations defining a frame, so-called *“frame cuts”*, many applications such as type inference and relation extraction are possible. For example, a subject-verb-object cut can be used to find the possible relations, which are the verb slots, between two entities, which are values of the subject and object slots, respectively.

ReVerb. Fader et al. argue that many open information extraction methods produce uninformative and incoherent extractions. Their ReVerb system [Fader et al., 2011] adds two syntactic and lexical constraints to the extraction of relation phrases in order to improve the quality of the results. To avoid incoherent and uninformative extractions, a syntactic constraint is introduced which checks if extractions match a predefined regular expression for the part-of-speech of each word. To further prevent overly specific extractions, a lexical constraint is added. This constraint checks each extraction against a dictionary which contains all relations for which at least 20 distinct arguments were found in a large document corpus. An evaluation of 500 test sentences shows that the ReVerb system outperforms several methods that were proposed earlier and do not use these constraints.

OLLIE. The OLLIE system further extends the scope of open information extraction [Schmitz et al., 2012]. Other than earlier systems, not only relation sentences that are mediated by verbs are considered, but also those mediated by other syntactic entities. Further, OLLIE is able to include contextual data, such as attribution or clausal modifiers into the extraction. The evaluation shows that this results in up to 146 times more correct extractions than are produced by ReVerb.

Summing up, the area of open information extraction is mainly concerned with the precise and scalable extraction of relations between named entities in large text corpora. To achieve this, different syntactic and lexical properties of unstructured, natural language text are exploited.

Approaches from Wrapper Induction

Early approaches to extracting structured data from the Web tried to build extraction programs, so-called “*wrappers*”, for individual web sites that would extract the data on their web pages. These methods then evolved into “*wrapper induction*” [Kushmerick et al., 1997] methods, which learn to create wrappers automatically from the structure of the HTML pages and human supervision. This is similar to the extraction of web tables, as the web table extractor can be interpreted as a very general wrapper that can be applied to every web site. The differences, however, are that wrapper induction methods are web site and topic driven, meaning that they create extractors for specific web sites and predetermined topics, while web table extractors are both web site and topic agnostic.

RoadRunner. Crescenzi et al. [Crescenzi et al., 2001] propose RoadRunner, a method for completely automatic wrapper generation. The method requires two web pages which are generated from the same web-page template as input and generates a generalised wrapper that matches both pages. This is achieved by comparing the web pages and generalising an initial, very specific wrapper for each mismatch. Specifically, the system chooses one of the web pages as initial wrapper, and then parses the second page using this wrapper. For every mismatch, i.e., difference between the two web pages, the wrapper is generalised. The authors note that some of the extracted datasets have a schema that is strongly influenced by the layout of the web pages, especially HTML tables, and would not be expected as a logical schema for the data. This indicates that wrapper induction methods cannot properly process web tables and that specialised methods, such as those presented in this thesis, are needed.

[Hao et al., 2011]. Hao et al. aim to extract structured data corresponding to known attributes of user-specified “*verticals*”, i.e., classes, from web sites. The method that they propose learns wrappers on one labelled web site, which can then be applied to other, unlabelled web sites. They propose three types of features to

match the text nodes in an HTML page to the provided attributes: layout features, content features and context features. Layout features use the visual position and arrangement of elements of the page when rendered in a browser. Content features describe the word tokens, character classes and content length of the text nodes. Context features measure the visual distance to other elements in the page as well as common prefixes and suffixes. From their experimental evaluation, the authors conclude that layout information is necessary for good performance as it complements the semantic information obtained from the content features.

Vertex. Gulhane et al. [Gulhane et al., 2011] present Vertex, a wrapper induction system developed at Yahoo! for high-precision information extraction. Similar to the approach proposed by Hao et al., Vertex requires a set of attributes and annotated web pages for each vertical that it should learn to extract. Further, the attributes are qualified with metadata that states if the attribute values are constant over time or must be part of every wrapper for this vertical. Before inducing wrappers, all web pages from a web site are clustered using shingle-based signatures [Broder et al., 1997] created from the HTML structure to identify different templates from which the web pages are generated. From these clusters, a set of web pages is sampled for human annotation and then used to learn the wrappers. The authors report that this system is deployed in production and used to extract more than 250 million records from more than 200 web sites.

DIADEM. Furche et al. [Furche et al., 2014] combine methods for web site exploration, identification of relevant data, and wrapper induction to create a system for “*automatic full-site extraction*”, called DIADEM. The system relies on a complex, manually created ontology that describes the topical domain that should be extracted. This ontology contains the domain schema including types and attributes, entity recognisers, and metadata such as mandatory attributes, modifiers, or specific metadata for HTML form filling. For a single domain, the creation of such an ontology supposedly takes 2-3 weeks time, but makes manual annotations on web pages unnecessary. The experimental evaluation of the system shows that the additional information stored in the extraction ontology enables better performance than achieved by earlier approaches.

The methods in the area of wrapper induction work either in an unsupervised or supervised fashion, with recent trends to use knowledge bases or specialised ontologies to create semi-supervised approaches. All methods rely on the common use of server-side templates for the generation of the web pages and exploit their regularities to extract the contained data. Other than web table extractors, wrapper induction systems are learned per web site or per class that should be extracted and are hence less generally applicable. However, once wrappers are learned from a web site, large amounts of information can be extracted which can then be used for knowledge base augmentation.

3.5 Conclusion

This section introduced knowledge bases as machine-readable representations of entities, their properties, and a taxonomy of classes in which these entities are organised. Further, common knowledge bases were described and compared. Then, additional details for DBpedia, the knowledge base that is used throughout this thesis, were given and statistics about the contents of this knowledge base were presented. Finally, the topic of knowledge base augmentation and its three tasks slot filling, schema extension, and entity-set completion were introduced.

The most commonly used knowledge bases in research that have been presented in this chapter differ in how they approach the organisation of their knowledge and the inclusion of additional knowledge. DBpedia and YAGO are based on completely automatic extraction from Wikipedia, and mainly differ in the size their class hierarchy and the number of used properties. While DBpedia uses a small, hand-crafted set of classes, YAGO bases its class hierarchy on WordNet, resulting in almost half a million different classes. In contrast to these two knowledge bases, Freebase and Wikidata rely on the contributions of end users to add additional statements to the knowledge base. Another frequently mentioned knowledge base is the Google Knowledge Graph, which supposedly is the large knowledge base at the time of writing, but due to its proprietary nature not much is known about its content or construction methodology.

The knowledge base that is used throughout this thesis is DBpedia. It contains a broad range of different entities of public interest due to its extraction from Wikipedia and is hence suitable for a general topical profiling of the contents of web tables. It is further recognised as a central reference point in the Web of Data, which supports its choice for the analyses in this thesis.

For the augmentation of knowledge bases, several tasks can be considered which extend the knowledge base with respect to missing values, properties, or entities. The discussed approaches from the areas of open information extraction and wrapper induction use methods that are specific to unstructured, natural language text or the hierarchical structure of elements in HTML documents. They are hence complementary to the methods for web tables, which are discussed in the remainder of this thesis.

Chapter 4

Web Tables

4.1 Introduction

The Web has undergone a rapid development in the past decades: from being a simple network of interlinked documents, it has grown to the largest platform for information and service providers that shape and impact our everyday life. But despite this tremendous progress, most content on the Web is still presented only in human understandable formats, and great effort is required to make this data also machine understandable. But this effort pays off, as it enables data-driven applications like search, question answering, or data mining to access the information on the Web more efficiently than possible for human users. Among many possible approaches in this area, such as natural language processing or promoting the semantic annotation of web pages, the use of HTML tables in web pages for data extraction promises to be a versatile source of information.

This chapter introduces the research area around web tables and discusses all necessary steps to extract a corpus, i.e., a large and structured collection, of web tables from HTML pages. This process needs to deal with a variety of challenges, such as distinguishing content tables from tables that are used for layout purposes or analysing the structure of content tables to recover their metadata, which is not available from their encoding in HTML. The focus in this and the following chapters is on relational web tables, but other types of web tables are also discussed in this chapter. Relational web tables contain data about a certain entity type and list multiple entities of this type with several attributes. This makes it natural to relate these web tables to and integrate them with other datasets, such as data in local spreadsheets [Yakout et al., 2012, Lehmborg et al., 2015] or the entities and properties in a knowledge base [Ritze et al., 2015].

To enable such applications, the web tables must be enriched with additional metadata by table understanding methods. Hurst [Hurst, 2001] conceptualised the process of table understanding as consisting of *table location and recognition*; *structural analysis*; and *interpretation*. For HTML tables, table location and recognition is relatively simple as the table and its rows and cells are identified by the

HTML mark-up code. This is not the case for tables that are extracted from images or scanned documents, which are, however, not in the scope of this thesis. For HTML tables it is, however, important to distinguish between such that are used to present data and such that are used to define the layout of elements on a web page. The structural analysis of tables comprises the recovery of table metadata, such as the location of headers or the data types of columns. Finally, semantic table interpretation is the process of annotating a table with elements that carry a semantic meaning, such as the classes, entities, and properties in a knowledge base.

This chapter focuses on the recognition and structural analysis of web tables. The first step is to classify the raw HTML tables along a taxonomy of table types, which includes types such as “*layout table*” or “*relational table*”. This is usually done by exploiting several structural and content based characteristics of the web tables in a supervised classification setting. The next step is to recover the metadata of the web tables. This starts with the detection of one or more “*header rows*”, which contain the names of the attributes that are stated in the web table. Such a detection can be performed by analysing regularities in the content of the table and differences in formatting. For example, the header of a column (which is usually a string value) that otherwise only contains numeric values can be detected by comparing the data types of the individual cells of a column. Such a data type detection is another part of the metadata recovery process, where data types of different granularity are assigned to every column of a web table. Methods in this area usually rely on the analysis of content patterns or use hand-crafted regular expressions. The last piece of metadata is the subject column, which is the column that states the names of the entities that are the main topic, i.e., the subject, of the web table. Knowledge about this column is the basis for many approaches that perform a semantic interpretation of web tables. It can be detected using measures such as the uniqueness of the columns or named-entity recognition methods.

All these individual steps accumulate to a quite involved pipeline of methods that need to be implemented before the semantic table interpretation of the data can even begin. This poses an entry barrier to the research on web tables. Research in this area is further hindered by the fact that most studies do neither publish their data, be it the original web pages or the extracted web tables, nor the implementation of the methods. To address these issues, the two large-scale web table corpora which are created during the work on this thesis, as well as the implementations of the used methods, are made publicly available.

The contributions of this chapter are:

- **Literature Survey:** This chapter introduces the web table extraction process and surveys existing literature along the individual steps of this process. The related work for each step is summarised and categorised according to the different heuristics and features that are used in the respective approaches. The approaches are further compared based on the results provided in their original publications, showing that the lack of public datasets limits their comparability.

- **Web Table Corpus Profiling:** This chapter gives an overview of the existing web table corpora that have been used in the related work. As most of these are non-public, only very limited information is available and the content and structure of the contained web tables is unclear. This situation is improved by the profiling and publication of two large-scale web table corpora. These corpora are publicly accessible, which allows researchers to work on the improvement of the state of the art without having to replicate earlier studies for the extraction of a corpus. The provided data profiles further show important characteristics of the web tables, which contain data of different data types and are mostly very small. As the following chapters will show, these characteristics have not been addressed sufficiently by existing work.

This chapter is organised as follows. Section 4.2 gives an overview of related work in the research area of web tables and presents common use cases. Section 4.3 then introduces the web table extraction process and its individual steps table classification, header row detection, data type detection, and subject column detection and discusses the relevant related work for each step. Then, Section 4.4 gives an overview of existing web table corpora and presents the two web table corpora that were created during the work on this thesis.

Parts of the work presented in this chapter, i.e., the data profiles of the two large-scale web table corpora, have previously been published in [Lehmberg et al., 2015, Lehmberg et al., 2016].

4.2 Related Work

This section gives an overview of the research on web tables and introduces the use cases for which web tables have been employed. A more detailed discussion of related work in the area of web tables for the individual topics of this thesis is provided in the following chapters.

Figure 4.1 shows an overview of the published literature on web tables between the years 2000 and 2018. Early work on web tables, such as the methods proposed by Chen et al. [Chen et al., 2000] and Wang et al. [Wang et al., 2000], was focused on the structural analysis of web tables. These early approaches considered the recognition and analysis of different types of tables and different, possibly very complex layouts of header rows and row labels. Then in 2008, Cafarella et al. [Cafarella et al., 2008b] presented the first approach that included applications using a large-scale corpus of web tables. In the following years, more research related to the semantic understanding of web tables [Limaye et al., 2010] and their applications [Yakout et al., 2012] was published and several use cases for the data from web tables were developed. These use cases in which web tables have been successfully employed as a data source are introduced in the following. These various possible applications show that web tables are an interesting and versatile source of information.

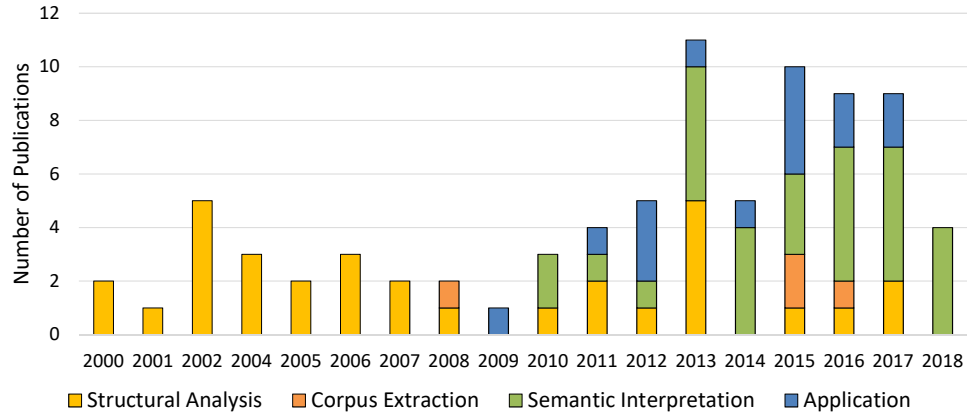


Figure 4.1: Overview of published research about web tables.

Search. Data from web tables can be used to enhance classical document search, or search engines can directly retrieve tables that match a user’s information need. For example, the works of Cafarella, Venetis, or Pimplikar [Cafarella et al., 2009, Venetis et al., 2011, Pimplikar and Sarawagi, 2012] enable users to search for data tables on the web. A publicly deployed service of this kind is the Google Fusion Tables service, which allows users to find and merge web tables [Balakrishnan et al., 2015].

Table Extension. If a user has a local data table and wants to extend it with additional rows or columns, web tables can be used to provide the additional data. Methods in this area [Cafarella et al., 2009, Yakout et al., 2012, Lehmberg et al., 2015] search for matching tables in a large corpus of web tables and then perform multiple joins to extend the local query table with the additional data.

Knowledge Base Augmentation. The data contained in web tables can also be used to construct or extend general-purpose knowledge bases such as DBpedia, Freebase or YAGO. Many authors, such as Zhang, Sekhavat, or Wang [Zhang, 2017, Sekhavat et al., 2014, Wang et al., 2012] have proposed methods to annotate web tables with elements of a knowledge base to perform a semantic interpretation of the data and to fill-in missing values in a knowledge base. This use case is the focus of this thesis, and the following chapters will discuss it in more detail.

Synonym Discovery. Cafarella et al. [Cafarella et al., 2008a] propose build a database of attribute co-occurrences in web tables, which can be used to suggest additional attributes to a user who is designing a relation. Others use the schemata of a large corpus of web tables in combination with a search engine query stream to discover attribute synonyms [Yakout et al., 2012, He et al., 2016].

Question Answering. Instead of semantically annotating web tables and integrating them with a knowledge base, it has also been proposed to use web tables directly for query and question answering. Such systems can, for example, be used to answer factual user questions, such as “*Britney Spears birthdate*” [Yin et al., 2011, Sun et al., 2016] or questions about quantities, such as “*height of Washington Monument*” [Sarawagi and Chakrabarti, 2014].

4.3 The Web Table Extraction Process

This section introduces the tasks and methods of table recognition and structural table analysis that are used in the web table extraction process. These include the detection and classification of web tables into different table types as well as the recovery of table metadata such as header rows and column data types. For each task, the commonly used heuristics and features are introduced, followed by a discussion of approaches that have been proposed in related work.

Other than web APIs or datasets in formats such as CSV or XML, web tables are published with the intention of a human consumer, i.e., are embedded in HTML documents which specify how the data, among other content, should be presented on a user’s computer screen. As a consequence, web tables cannot be accessed directly, for example via URLs or file names, but need to be extracted from HTML pages first. This means, the only way to get access to web tables is to load a web page, interpret the HTML mark-up, check if a web table is present, and then extract its data. As a consequence, web tables also cannot be interlinked on the Web, which means that the only way to create large collections of web tables is to crawl large numbers of web pages.

After the crawling phase, the web tables can be extracted from the HTML documents. However, not all HTML `<table>` tags represent tables that contain data. In fact, most occurrences of this tag are used for layout purposes and it is necessary to use classification methods to distinguish between the different possible types of tables. On the structural level, the HTML mark-up code specifies how the data in a web table is organised in a two-dimensional grid of rows and columns, but does not contain any other metadata, such as data types of the columns. To enable downstream applications to interpret the data, these and other types of metadata must hence first be recovered.

The rest of this section describes these steps of the web table extraction process:

- **Table Classification:** Distinguishes between tables that are used as layout elements and different types of tables that contain factual data.
- **Header Row Detection:** Discovers the headers of the columns in the table.
- **Data Type Detection:** Discovers the data types of the columns in the table.
- **Subject Column Detection:** Discovers the column that contains the names of the subject entities that are described in the table.

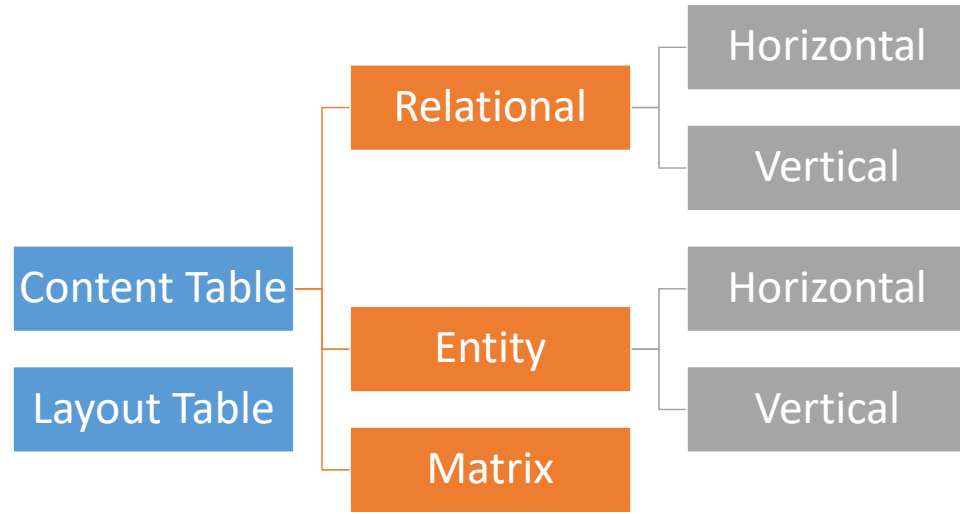


Figure 4.2: Table Type Taxonomy

4.3.1 Table Classification

This section introduces the table classification task. The goal of this task is to assign a table type to a given HTML table. These table types allow to distinguish between tables used for layout purposes and different types of tables which contain factual data. This section first introduces a taxonomy of table types, and then presents heuristics and supervised classification approaches for this task.

Tables can be detected in HTML documents by their corresponding HTML tag `<table>`. However, not all occurrences of this tag indicate the presence of tabular data. In fact, most of the time this tag is used for layout purposes (97.5-98.9% of all cases [Cafarella et al., 2008a, Crestan and Pantel, 2011]). Among the tables which actually contain data, there are different table layouts which need to be considered when interpreting the data. For example, relational tables can be formatted horizontally (columns correspond to attributes) or vertically (rows correspond to attributes).

It is hence necessary to classify the detected HTML tables into a taxonomy of table types before further processing. Such taxonomies have been proposed by Crestan and Pantel [Crestan and Pantel, 2010, Crestan and Pantel, 2011] as well as Lautert et al. [Lautert et al., 2013]. In most practical applications, however, only the distinctions between layout tables and content tables in general, as well as between relational tables and entity tables for content tables are used. Figure 4.2 shows the taxonomy of table types that will be used in this thesis and Figure 4.3 shows examples for different types of content tables in this taxonomy.

Relational tables (Figure 4.3a) contain data about several different entities and contain multiple attributes of these entities, similar to tables in a relational database. Entity tables (Figure 4.3b) contain data about only a single entity, which is often not stated in the table, and usually list a larger number of attributes. Both relational

	Lake	Area
1	Windermere	5.69 sq mi (14.7 km ²)
2	Kielder Reservoir	3.86 sq mi (10.0 km ²)
3	Ullswater	3.44 sq mi (8.9 km ²)
4	Bassenthwaite Lake	2.06 sq mi (5.3 km ²)
5	Derwent Water	2.06 sq mi (5.3 km ²)

(a) Relational Table

Government^[3]	
• Type	Mayor–Council
• Body	New York City Council
• Mayor	Bill de Blasio (D)
Area^[2]	
• Total	468.9 sq mi (1,214 km ²)
• Land	304.8 sq mi (789 km ²)
• Water	164.1 sq mi (425 km ²)
• Metro	13,318 sq mi (34,490 km ²)
Elevation^[4]	33 ft (10 m)

(b) Entity Table

	Right-handed	Left-handed	Total
Males	43	9	52
Females	44	4	48
Totals	87	13	100

(c) Matrix Table

Figure 4.3: Examples of different table types.

and entity tables can have a horizontal or a vertical layout, in which the meaning of rows and columns is interchanged. Finally, matrix tables (Figure 4.3c) have more than one dimension of attributes and often state them both in the first row and first column of the table.

To classify web tables into such a taxonomy, usually a two-step approach is applied. First, heuristics filter out the large amounts of layout tables, typically removing 80 - 90 % [Crestan and Pantel, 2011, Cafarella et al., 2008a] of all detected tables. Then, a supervised classification method is used to classify the remaining tables into some of the table types introduced earlier. An extensive discussion of the used methods can be found in the survey by Zanibbi et al. [Zanibbi et al., 2004].

Table Classification Heuristics

This section introduces heuristics which have been proposed to distinguish between layout tables and content tables and discusses their effectiveness as reported in the literature. Due to the large volume of tables which need to be processed during the extraction from a web crawl, computationally efficient heuristics are often used in a first step to filter out obvious non-content tables before applying a more complex classifier in a second step. In the following, the commonly used heuristics are presented.

Table Size. Very small tables are considered to be meaningless and hence discarded. For example, a table with only a single row or a single column cannot contain any relational data. Such tables are often used to align other elements on the web page, i.e., they are layout tables. Usually, a minimum number of rows and a minimum number of columns are required [Chen et al., 2000, Penn et al., 2001, Cafarella et al., 2008a, Crestan and Pantel, 2011, Eberius et al., 2015].

Nesting. Tables which contain other tables in their cells are considered to be layout tables. Although tables with complicated layouts, which use nested column headers or drill-down rows, exist, they can be achieved with the use of column and row spanning. Placing a table inside another table is rather used to achieve a grid-like layout on the web page via the outer table. Hence, only the innermost tables are kept [Penn et al., 2001, Wang and Hu, 2002].

Table Content. Another indicator that a table is used for layout purposes is the content length of cells. In a data table, short strings or numbers are expected, while layout tables contain longer texts or mostly hyperlinks. Tables with cells which exceed a certain length [Crestan and Pantel, 2011, Penn et al., 2001] or mostly contain certain HTML tags [Penn et al., 2001, Chen et al., 2000] can hence be filtered out.

The heuristics introduced above have been applied in several studies to filter out a large part of the obvious non-content tables. Wang and Hu [Wang and Hu, 2002] report the lowest amount of removed tables with only 22% of 14 thousand tables in their dataset using the nesting heuristic. Cafarella et al. [Cafarella et al., 2008a] exclude all web tables with fewer than two rows or columns, web tables which are embedded inside HTML forms, and web tables which represent a calendar. They report to filter out 89% of 14.1 billion web tables using these heuristics. Crestan and Pantel [Crestan and Pantel, 2011] use the nesting heuristic, a minimum table size of two rows and two columns and require that no cell is longer than 100 characters. With these heuristics they remove 84% of their 8.2 billion web tables and verify the quality of the heuristic through the evaluation of 200 randomly selected tables, which shows that 93% of the removed tables are actually layout tables. Eberius et al. [Eberius et al., 2015] only use the table size heuristic with a minimum of two rows and two columns and filter out 92% of the 26 thousand tables in their dataset and Wang et al. [Wang et al., 2012] use the nesting, table size, and table content heuristics and report that 3.4% of 1.95 billion tables remain after filtering.

Supervised Table Classification

After the heuristic filtering, a supervised classification approach is used to classify the remaining tables into two or more different table types. This section first introduces the different types of features that have been proposed for this task, and then discusses related work that uses these features to create a supervised classification model. The classification models use combinations of the different types of features which are introduced in the following.

Global Features. Global features capture characteristics of the whole table, such as the maximum, mean, and variance of the number of rows, columns, and cell length [Crestan and Pantel, 2011, Wang and Hu, 2002, Cafarella et al., 2008a, Eberius et al., 2015]. Here, the same intuition as for the heuristics is applicable: relational tables should have rather short cell values and be more consistent over all cells, while layout tables can contain cells with large amounts of contents or completely empty cells.

Local Features. Local features are calculated for each column and row individually. Similar to global features, they capture cell length mean and variance as well as the ratio of spanning cells [Crestan and Pantel, 2011]. If a table is relational, all values for the same attribute should be consistent in length and so should be the cells of each column. Eberius et al. [Eberius et al., 2015] additionally calculate the frequency of several HTML tags and special characters as local features.

Structural Features. A different approach to capture the structure of tables is taken by Son and Park [Son and Park, 2013], who use parse tree kernels [Collins and Duffy, 2002, Zhang and Lee, 2003] to transform the HTML parse tree of a table and its context into feature vectors. This allows them to detect semantically identical tables which are syntactically different, i.e., the tables contain same values but their HTML representation contains, for example, additional tags.

Content Type Features. Content type features describe which types of content are encountered in the table. These features are calculated as the frequency of cells with content types such as image, form, hyperlink, alphabetical, digit, empty, or other [Wang and Hu, 2002, Cafarella et al., 2008a, Eberius et al., 2015]. Additionally, an average content type consistency can be calculated per row and column [Wang and Hu, 2002, Eberius et al., 2015]. Two subcategories of this group of features are HTML features and lexical features [Crestan and Pantel, 2011]. HTML features are the relative frequency of certain HTML tags, such as a, img, input, etc., and lexical features show the relative frequency of special characters or numbers in the cells of the table. Wang and Hu [Wang and Hu, 2002] also use a Word Group feature, which uses a vector space representation of the text in the table and word clustering to calculate similarities with the tables seen during training based on the word tokens in the table.

Classification Approaches

Different combinations of the features introduced in the section above were used in several studies. However, due to the lack of a publicly available benchmark dataset, the comparability of these studies is limited. Most of the studies consider the binary classification task `content table` vs. `layout table` and report the performance of their method on non-public datasets, which were manually annotated by the authors of the respective studies. In the following, an overview of each study and its findings is provided.

[Wang and Hu, 2002]. Wang and Hu use global and content type features to train a binary decision tree classifier. The evaluation is performed on a hand-labelled dataset of 14 609 tables collected from 1 393 HTML pages using 9-fold cross validation. The authors find that content type features perform better than global features if used individually, but the combination of both results in the best performance of 96% F1-measure for the content table class. They further compare their approach to the purely heuristic method proposed by Penn et al. [Penn et al., 2001], which only achieves an F1-measure of 62% (88% after slight modifications) on the same dataset. The features proposed by Wang and Hu have later been used for similar classification approaches by Jung and Kwon [Jung and Kwon, 2006] and Cafarella et al. [Cafarella et al., 2008a], but neither of the approaches was shown to outperform the initial study.

[Crestan and Pantel, 2011]. Crestan and Pantel present a classifier to differentiate between the 10 different table types which they propose in their taxonomy and hence solve a more fine-grained and harder classification task. The evaluation is performed on a dataset of 5 000 randomly sampled tables, which were annotated by 10 paid editors, using 20-fold cross validation. Using gradient-boosted decision trees, they report F1-measures between 24% (horizontal listing) and 91% (calendar) for the 10 possible classes. On the simpler binary classification task, their method achieves an F1-measure of 68% for the content table class, which outperforms the approach of Penn et al. [Penn et al., 2001] (49%) and another classifier which uses the features proposed by Wang and Hu [Wang and Hu, 2002] (55%).

[Son and Park, 2013]. Son and Park propose a parse tree kernel to transform the HTML structure into features for a binary SVM classifier. They are the first to propose to use the HTML document structure for feature generation. The evaluation is performed on the dataset created by Wang and Hu [Wang and Hu, 2002] using 10-fold cross validation. The combination of the parse tree kernel and the content features proposed by Wang and Hu results in an F1-measure of 99% for the content table class, compared to 96% achieved by the method proposed by Wang and Hu, and 62% achieved by the heuristics proposed by Penn et al.

[Eberius et al., 2015]. Eberius et al. combine global, local, and content type features and evaluate several different classifiers. They consider the table types layout, vertical listing, horizontal listing, matrix and other as well as the binary classification problem. Using a correlation-based feature selection [Hall, 1999] they determine 31 relevant features from the initial 127 features that they proposed. For the binary classification task, the ratio of cells containing images or forms as well as `<th>` and `` tags is more important than other features, while the average cell, column, and row sizes are more relevant for the differentiation between different content table types. The evaluation is performed on a manually annotated dataset of 2 022 tables using repeated random sub-sampling, where the dataset is randomly split into 90% training and 10% test data over 100 iterations. Using a Random Forest classifier, they achieve F1-measures between 60% (Other) and 95% (Layout). For the binary classification task, they report F1-measure values between 89% (SVM) and 91% (Random Forest).

Many different approaches for the detection of content tables as well as for the differentiation of table types have been proposed and the consensus in the literature is to use features that capture many different aspects of the web tables and learn a supervised classification model for the task. The used datasets and methods are often not publicly available, but seem to be shared in private communication so a comparative evaluation is possible. The usual performance for the detection of content tables is very high, with F1 scores which are often above 90%, but for a more detailed classification the performance varies from type to type between 24% and 95%.

4.3.2 Header Row Detection

This section introduces the header row detection task. First, the heuristics and features as proposed in the literature are introduced. Then, approaches from the related work and their evaluation are discussed.

A header row contains the column headers and states the names of the attributes that are represented by the columns in a table. These names are metadata and need to be treated differently from the content data of the table, as they often contain clues to the semantic meaning of the column's content. Hence, it is necessary to determine which rows of a table are header rows and which ones are data rows. The designated HTML tag to indicate header rows, `<th>`, which would make this detection trivial, is only used in 20% of the content tables [Pimplikar and Sarawagi, 2012]). Thus, the use of header row detection methods is required to determine which row(s), if any, contain column headers.

Features and Heuristics

For the detection of header rows, different types of features and heuristics are applied, which range from simple positional assumptions, such as “*the first row is the header row*”, to the analysis of the formatting and content of different cells in a table. The following introduces the different types of features and heuristics that have been proposed in the literature.

Position. The simplest heuristic is to assume that the first row is the header row, which is the usual design of a horizontal content table [Pinto et al., 2002]. A more advanced version of this heuristic is used by [Cafarella et al., 2008a], who learn a classifier that detects whether or not a table contains a header. If yes, they always choose the first row to be the header row.

Formatting. Header rows are often formatted differently from the rest of the table to provide visual clues to the human users. Many approaches hence check for differences in background colour or font formatting between successive rows in the table [Jung and Kwon, 2006, Pimplikar and Sarawagi, 2012, Limaye et al., 2010, Wang et al., 2012].

Content. If the table contains non-string values, header rows can be detected by the change of data type. For example, all cells in a column representing the age of people contain numbers, except for the column header which contains the string “*age*” [Jung and Kwon, 2006, Wang et al., 2012]. Other approaches try to use the values in the cells to either learn probabilities for the value being a column header [Yoshida et al., 2001], or to look up the values in external resources such as a database of attribute names [Cafarella et al., 2008a] or a knowledge base [Wang et al., 2012].

Header Row Detection Approaches

After the introduction of the different types of features and heuristics, this section now discusses the approaches that apply them for header row detection. Similar to the table type classification task, many approaches report their performance for the header row detection task, but the lack of a common benchmark dataset hinders an objective comparison.

[Jung and Kwon, 2006]. Jung and Kwon use several formatting and content features to calculate a score for each cell. They then apply a threshold to decide which cells represent a header, i.e., their approach can annotate a mixture of rows and columns as headers. The used features check for the presence of a `<th>` tag, differences in background colours, font attributes, empty or spanning cells and content type changes. The method is evaluated on a dataset of 2 565 web tables and achieves an accuracy of 82%.

[Cafarella et al., 2008a]. Cafarella et al. use a rule-based binary classifier to decide whether a table has a header in the first row or not. The used features are similar to the global and local features used in table type classification. As a second approach, they propose to incorporate information from a database of attribute co-occurrence statistics, which they generated from their table corpus. Both approaches are evaluated on a dataset of 1 000 tables using 5-fold cross validation. The approach using global and local features achieves an F1-measure of 81% and the approach that additionally uses attribute statistics improves this result to 87%.

[Pimplikar and Sarawagi, 2012]. Pimplikar and Sarawagi use formatting and content features to detect header rows. Their method iterates over all rows of a table and marks them as header row if they are different from most of the remaining rows. They do, however, not elaborate how difference is measured. An evaluation of this method on a dataset of 2 000 web tables shows an accuracy of 99.8%. They further report that in their corpus of 25 million web tables, 60% of the tables have one header row, 18% have no header, 17% two header rows, 5% more than two header rows.

Approaches for the detection of header rows often use rather simple heuristics or the methods are only superficially described in the respective publications. This indicates that the task is not very challenging or applications following a header row detection are resilient to noise. The most common approaches use the position or formatting of rows to decide if a row is a header row.

4.3.3 Data Type Detection

This section introduces the data type detection and unit detection tasks. First, the literature on data type detection is presented, followed by related work on unit detection.

The values in the cells of a web table are encoded as strings in the HTML mark-up code and do not contain any indication of the data type. This is problematic for non-textual values, which can be represented in a variety of different formats. For example, the number 1 000 can be represented as “1,000.00” or as “1.000,00” and the date “1st of February 2019” can be represented as “1/2/19” or as “2/1/19”. For a correct interpretation of these values, it is hence necessary to detect the appropriate data type, such as `numeric`, `date`, or `string` and to parse their formatting correctly.

Compared to the tasks of table type classification and header row detection, only few approaches for data type detection exist. This is probably due to the fact that many approaches for semantic table interpretation are limited to columns that contain named entities, and hence have no need to determine the data type of other columns. Such named entity columns are a subset of columns with the `string` data type, and approaches for semantic table interpretation detect which type of entities the columns correspond to (see Chapter 5).

[Kim and Lee, 2005]. Kim and Lee propose to detect 15 different data types with predefined textual patterns and keywords. The presented patterns capture different data types with specific formats, such as postal codes or dates and are a mixture of HTML types (`image`, `form`), date types (`time`, `date`, `month`, `day`), primitive types (`string`, `number`, `blank`), units (`temperature`, `voltage`, `weight`, `currency`, `percentage`) and semantic types (`postal code`). No evaluation of the data type detection method is reported.

[Zhang, 2017]. Zhang determines the data type of columns by checking regular expressions for each cell. Based on the matches, one of the data types `empty`, `named entity`, `number`, `date`, `long text`, and `other` is assigned to each cell. The final data type for a column is then determined by majority voting among all cells of the column. Again, no evaluation of the method is reported.

The detection of data types has not received much attention in the literature, and is often not even evaluated. This is interesting, as different data types can lead to considerably different interpretations of data. A possible reason is the focus of many applications on named entities, for which columns of other data types are not useful.

Unit Detection

The task of unit detection is related to data type detection, as it is a more fine-grained categorisation of numeric columns. Detecting units allows for the transformation of values into a common base unit to make them comparable. For example, values stated in miles and kilometres can be converted to metres.

[Hignette et al., 2007]. Hignette et al. distinguish between numeric and symbolic columns using a set of hand-crafted rules. A cell is classified as numeric if it contains a number, a number followed by a unit, or contains more numbers and units than words. If it contains more words than numbers and units it is classified as symbolic. The type of a column is determined by majority voting among the cells of that column. The method is evaluated on a dataset of 60 tables taken from publications in food microbiology, showing an accuracy of 98%.

[Zhang and Chakrabarti, 2013]. Zhang and Chakrabarti use a user-defined set of unit conversion rules and a lookup mechanism to assign candidate units to numeric columns. A joint inference over a graph of semantically similar columns using a probabilistic graphical model is then used to propagate the unit annotations among the columns. The method is evaluated on 1000 randomly selected numeric columns that represent company revenue. The authors find that the graphical model creates more annotations than the lookup-based method, with a precision of 97%.

[Sarawagi and Chakrabarti, 2014]. Sarawagi and Chakrabarti collect a unit catalogue from Wikipedia with 44 quantity types and 750 units, each associated with one or more full names, symbols, and lemmas. They then evaluate two different unit parsers. The first one is a simple rule-based parser which uses the names, symbols, and lemmas to match units in column headers. The second parser uses a context-free grammar and several features to score each possible production of the grammar. An evaluation on a dataset of 617 tables shows 40% accuracy for a rule-based extractor and 82% accuracy for the parser using the context-free grammar.

Compared to the few and rather simple approaches on general data type detection, units have received much more attention. A reason can be that there are clear use cases for answering numerical queries that cannot be solved without converting the values from different sources into a common base unit. The most successful approaches for the detection of units in web tables use a collective inference over multiple web tables and their possible interpretations.

4.3.4 Subject Column Detection

This section introduces the subject column detection task. A subject column is a pseudo-key of a web table and important for its semantic interpretation. Most approaches for semantic table interpretation work under the assumption that each relational web table contains one column which states the name of the entities which are the subject of the rows. Although many methods treat it like a key of the web table, it is not necessarily unique, i.e., the subject column generally is only an approximate key. The following paragraphs describe the heuristics and other approaches that have been proposed for this task.

Heuristics. Positional heuristics have been proposed by [Pinto et al., 2002, Cafarella et al., 2008b], who interpret the left-most column as subject column. An extension of this heuristic was proposed by [Venetis et al., 2011], who use left-most column which is neither of type `numeric` nor `date`.

[Venetis et al., 2011]. Venetis et al. propose a binary SVM classifier, which decides for each column whether it is a subject column or not. If multiple subject columns are detected, the column with the highest prediction confidence is chosen. The used features are the fraction of cells with unique or numeric content, the variance in the number of date tokens in each cell, the average number of words in each cell, and the column index from the left. An evaluation on a test set of 200 tables, which all contain a subject column, shows that the classifier beats a heuristic which always chooses the left-most string column with an accuracy of 94% vs. 83%. For another dataset which contains 160 randomly selected tables, of which only 97 have a subject column, the authors report an F1-measure of 72%.

[Wang et al., 2012]. Wang et al. use lookups in the Probase knowledge base to find subject columns. The lookups are performed for all cells in the same column to find matching entities and for the headers of all other columns to find matching attributes.. The column with the largest intersection of the concepts of matched entities and attributes is chosen as subject column. The approach is evaluated a set of 189 randomly selected tables from Wikipedia, showing an accuracy of 87%.

[Zhang, 2017]. Zhang proposes an unsupervised method that scores each column using a linear combination of several features. Only columns with the data type `named entity`, which is determined using regular expressions, are considered. The used features are calculated based on the fraction of empty or unique cells, the distance from the left-most named-entity column, the presence of acronyms or IDs, the frequency of the column header in the table’s context, and the number of matches in a web search. The features are combined with hand-crafted weights into a final score and the column with the highest score is chosen as subject column. However, Zhang does not report an evaluation for this approach.

Although the subject column is the most important column of a web table for many semantic table interpretation approaches, the methods to detect this column are rather simple. Common approaches either use features which are similar to those used in header detection, the position and uniqueness of columns, or the presence of entity names known from a knowledge base.

4.4 Web Table Corpora

This section compares the statistics that have been published about different corpora of web tables that have been used in the literature. This comparison shows that despite many publications in this area, only little information is publicly available and that there is a need for openly available corpora. Afterwards, the two public, large-scale web table corpora that were created during the work on thesis are described and their data profiles are presented.

Although a large number of studies have investigated the web table extraction problem, there are only very few that actually apply their methods on a large scale. Until 2014, when the first Web Data Commons Web Tables Corpus (WDC WTC 2012, see Section 4.4.1) was released, no public, large-scale corpus of web tables did exist. This clearly hinders research on the topic, as researchers need to replicate table extraction, classification, and metadata recovery methods before they can start working with the tables.

Table 4.1 shows an overview of the existing large-scale web table corpora that have been mentioned in the related work. All statistics are reproduced as stated in the original publications and a “-” indicates that the statistic was not provided. The column “*Pages*” states the number of HTML pages in the used web crawl, “*Tables*” refers to the number of HTML tables extracted from these pages using

filtering heuristics, and “*Content Tables*” shows the number of non-layout tables after a classification step. The “%” column indicates the percentage of HTML tables which are content tables. Of all the corpora mentioned in the table, only the last three are publicly available.

Figure 4.4 further shows a comparison of the distribution of web tables over the number of rows and number of columns for the corpus created by Cafarella et al. [Cafarella et al., 2008a] and the WTC 2012 and 2015. All three corpora agree on the distributions, which show that the majority of all web tables is small, with only 2-9 rows or columns.

Table 4.1: Overview of web table corpora (K=thousand, M=million, B=billion).

Corpus	Pages	Tables	Content Tables	%	Public
[Wang and Hu, 2002]	1.4K	14.6K	1.3K	9.0%	no
[Cafarella et al., 2008a]	>1B	14.1B	154M	1.1%	no
[Yin et al., 2011]	-	-	744M	-	no
[Venetis et al., 2011]	-	-	12M	-	no
[Crestan and Pantel, 2011]	1.2B	8.2B	1.3B	15.9%	no
[Wang et al., 2012]	1.68B	1.95B	69M	3.5%	no
[Pimplikar and Sarawagi, 2012]	500M	-	25M	10.0%	no
[Yakout et al., 2012]	-	-	573M	-	no
[Sarawagi and Chakrabarti, 2014]	500M	-	25M	-	no
WDC WTC 2012	3.5B	11B	147M	1.3%	yes
[Eberius et al., 2015]	3.6B	-	125M	-	yes
WDC WTC 2015	1.78B	10B	233M	2.3%	yes

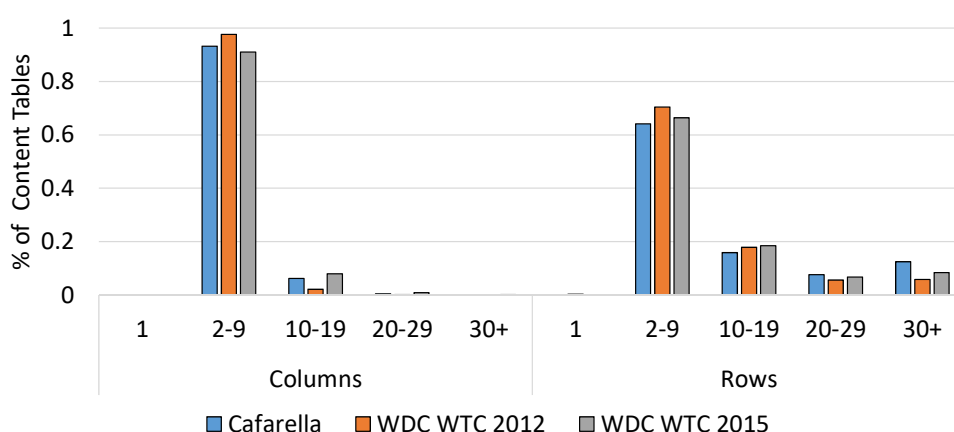


Figure 4.4: Comparison of the distributions of web tables by rows and columns over different corpora.

4.4.1 Web Data Commons Web Tables Corpus 2012

This section presents the Web Data Commons Web Tables Corpus¹ (WTC) 2012, which is the first non-commercial, large-scale corpus of relational web tables that is publicly available. It is based on a public web crawl that is provided by the CommonCrawl Foundation². Specifically, the corpus is extracted from the 2012 version of the Common Crawl, which contains 3.5 billion HTML pages from 43 million different web sites. The following sections describe the table extraction and metadata recovery steps that are applied during the creation of the corpus and then present a data profile of the extracted web tables.

Extraction Process

This section describes the extraction process that was executed to create the Web Data Commons Web Tables Corpus 2012. This extraction process follows the steps described in Section 4.3 and classifies all detected HTML tables into different table types and then recovers different types of metadata for all tables that were classified as content table.

First, the HTML pages are parsed and all HTML table tags are considered as potential candidates for extraction. These candidates are filtered using the nesting heuristic, i.e., only tables which do not contain other tables are kept, and the table size heuristic, which filters out all tables with less than 5 cells or less than 3 rows. This filtering results in 11 billion web tables, on average 3.4 per web page.

Next, a binary classifier is used to distinguish between layout tables and content tables. This classifier uses local features and content type features, as described in Section 4.3.1. Specifically, the used local features are average and standard deviation of column, row, and cell length as well as the average cumulative length consistency. The average cumulative length consistency measures the consistency of the cell content length within the rows and columns of the web table [Wang and Hu, 2002]. The used content features are percentage of link, form, and image tags, percentage of empty, numeric, and textual cells as well as the average content type consistency. The average content type consistency measures the consistency of data types of the cells within the rows and columns of a web table [Wang and Hu, 2002]. Based on these features, a decision tree classifier is trained, which achieves a precision of 58% and a recall of 62% for relational tables on a test set of 77 630 web tables collected from 7 350 randomly selected web pages [Lehmberg et al., 2015].

This classifier is applied to the 11 billion innermost tables, resulting in 147 million tables (1.3%) which are classified as content table. This is in line with the results of Cafarella et al. [Cafarella et al., 2008b], who reported that 1.1% of all web tables contained relational data. For these 147 million relational web tables, the metadata recovery process, as described in the next section, is applied.

¹<http://webdatacommons.org/webtables/>

²<http://commoncrawl.org/>

On a technical level, the HTML tables which are classified as content tables are transformed into CSV files. For this transformation, every row of an HTML table as marked-up through the `<tr>` HTML tag is converted into a line and all cell values as marked-up through the `<td>` HTML tag are separated by comma. Occurrences of the `cellspan` or `rowspan` HTML attribute are handled by repeating the same value for all affected cells. Missing cells in the HTML code are filled with empty values.

Metadata recovery

This section describes the different types of metadata that are provided for the web tables in the corpus as well as the methods used to recover these metadata. All methods are variations of the methods described in Section 4.3.

Header Row Detection. To determine the header rows of the web tables, the position heuristic is applied. This heuristic annotates the first row as header row if at least 80% of its cells are non-empty. An evaluation of this method on 1 000 randomly selected tables shows an accuracy of 82%.

Data Type Detection. The data type detection method first assigns one of the data types `string`, `numeric`, `date`, `boolean` or `list` to each cell in the table. Then, a majority voting of all cells in a column determines the final data type. The assignment of data types to cells is performed by checking a set of regular expressions for each data type in a specific order. First, if a value is marked as a list by the table extractor, which happens in cases where the cell contains an HTML list tag (`ul` or `ol`), the type `list` is assigned. Otherwise, regular expressions for `numeric` values with units are checked, then `boolean` values, followed by `date`, and finally `numeric` values without unit. If no match was found for these data types, the value is marked as type `string`. Numeric values with unit are treated differently than numeric values without unit by using a separate set of regular expressions that can detect various units of measurement as well as their abbreviations and unit symbols. The cell values for columns with this data type are converted to a pre-defined base unit, e.g. miles and kilometres are converted to metres. An evaluation of this method on 1 000 randomly selected tables shows an accuracy of 89%.

Subject Column Detection. For the detection of the subject column, multiple heuristics are applied. All columns with data type `string` and an average cell length between 3.5 and 200 characters and a uniqueness above 0.3 are considered as candidates. The uniqueness is defined as the ratio of unique values u to all values v of a column, reduced by the ratio of missing values m : $\text{uniqueness} = \frac{u-m}{v}$. This uniqueness score rewards columns with many unique values and penalises columns with many missing values. After the candidate generation, two rules decide which

column is selected as subject column: (1) if one of the candidates has the column header “*name*” or “*title*”, it is chosen; (2) otherwise, the column with the highest uniqueness score is chosen. In case of a tie, the left-most of the tied columns is chosen. An evaluation of this method on 1 000 randomly selected tables shows a precision of 76% and a recall of 84%. Note that precision and recall are used for evaluation here instead of accuracy, as not every web table has a subject column.

Data Profiling

This section presents a data profile of the web tables contained in the extracted corpus. This profile shows the distributions of table size, column data types, and origin with respect to the top-level domain of the web pages from which they were extracted.

Table Size. Figure 4.5 shows the frequency distribution of web tables with respect to their size in terms of rows and columns. For rows, only data rows are considered, i.e., the header rows are not included in the statistic. The two series labelled “*x Rows*” and “*x Columns*” indicate the relative frequency of web tables with exactly as many rows or columns as indicated by the horizontal axis. The two series labelled “*At least x Rows*” and “*At least x Columns*” indicate the relative frequency of web tables with at least as many rows or columns as indicated by the horizontal axis (complementary cumulative frequency).

The distributions show that web tables tend to be small. Only 25% of all tables have more than ten rows and only 2% of all tables have more than ten columns. The mode of both distributions is 2, i.e., the largest number of web tables is found for two rows or columns, with 15% of all tables for rows and 48% of all tables for columns. The median is 6 for rows with an average of 12.41 and 3 for columns with an average of 3.49.

It can be assumed that designers try to keep their tables small, such that they fit on a user’s computer screen, and hence limit the number of columns. There is, however, more tolerance concerning the number of rows: while only 2% of all web tables have more than ten columns, 24% of them have between eleven and one hundred rows. This distribution of web tables over rows could indicate frequent use of *paging*, where a table is split into several parts and shown over multiple pages.

Data Types. Figure 4.6 shows the distribution of web table columns over the detected data types. Most columns are either of type `string` (65%), which includes named entity columns, or `numeric` (28%). The other types `date` (3%), `boolean` (2%) and `list` (2%) have only been detected for very few web tables. While this distribution shows a dominance of the `string` data type, 35% of all columns are non-textual. Most of these non-textual columns are numeric, which includes numbers with and without units.

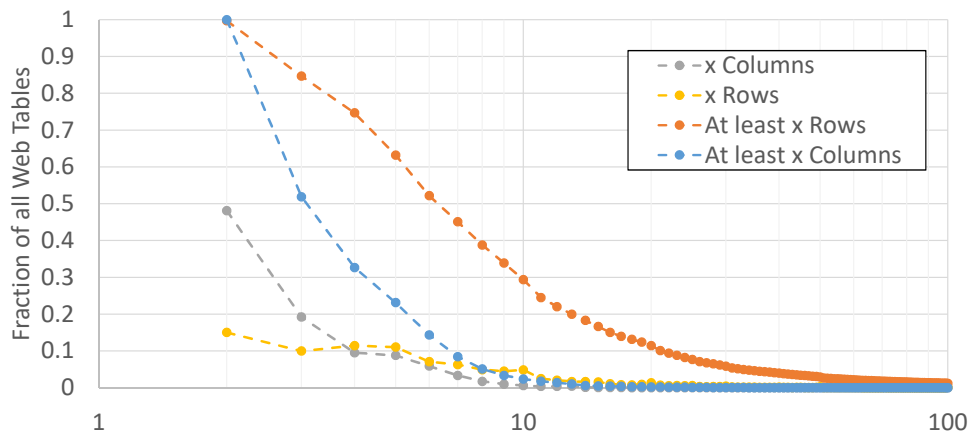


Figure 4.5: WTC 2012: Distribution of web tables over rows and columns.

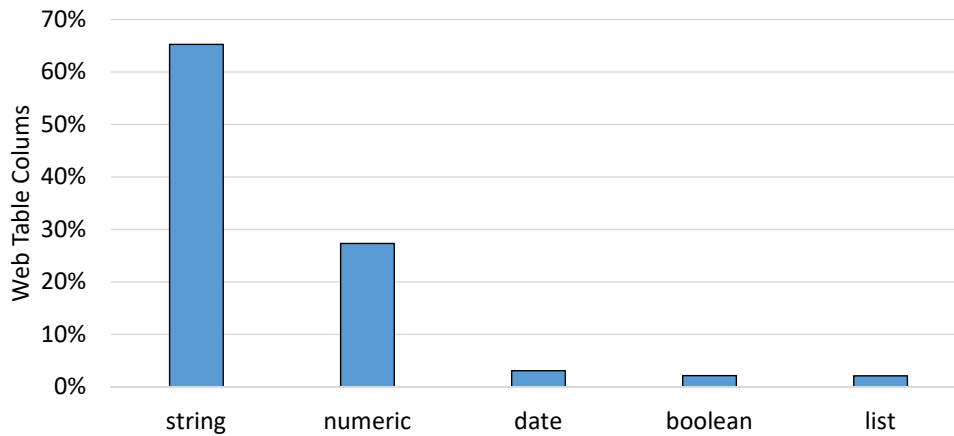


Figure 4.6: WTC 2012: Distribution of web table columns over data types.

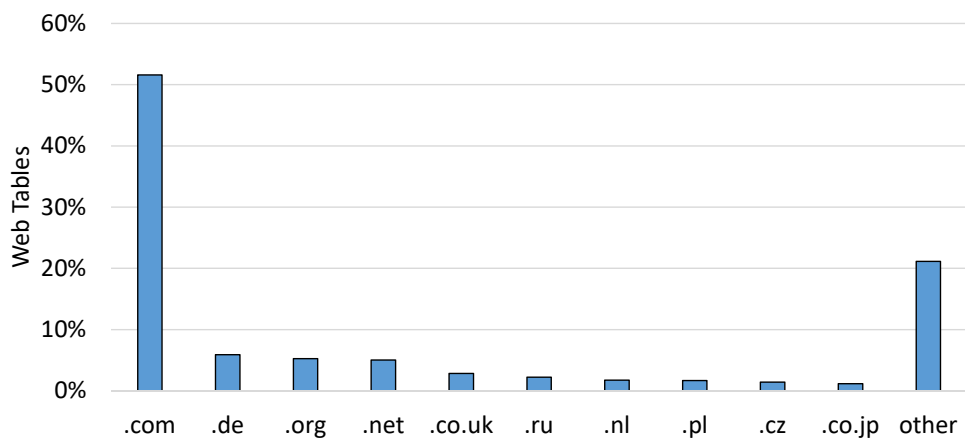


Figure 4.7: WTC 2012: Distribution of web tables over top-level domains.

Column Headers. More insights into the topics of the web tables can be seen in the frequency distribution of column headers. Among the 50 most frequent column headers, the most frequent group consists of generally applicable words that don't reveal the topic of the table such as “*name*”, “*date*”, or “*rank*”, which occur for 4% of all columns in the corpus. The next-largest group is shopping-related with headers such as “*price*”, “*model*”, “*use from*”, or “*new from*” that make up another 2% of all columns. Another frequent group of headers actually represents data values (“1”, “2”, etc.) and accounts for almost 2% of all columns. The most frequent column header is, however, an empty string for 9% of all columns.

Table Origin. The origin of the web tables in the corpus can be analysed via the top-level domain (TLD) of the URLs of the web pages that contains the web tables. Top-level domains are designated for specific purposes, for example .com for commercial web sites and .org for organisations. Further, country-code top-level domains such as .de or .co.uk indicate the country of origin of a web site. Figure 4.7 shows the distribution of web tables over the ten most frequent TLDs. The majority of all web tables originates from web pages with the .com TLD (52%), followed by .de (6%) and .net (5%). This indicates that the majority of web tables is provided by commercial web sites and might be related to products or services that are offered. This distribution is expected and very similar to the distribution of all web pages in the used crawl over TLDs with 49% for .com, 7% for .de and 5% for .net.³

4.4.2 Web Data Commons Web Tables Corpus 2015

This section presents the Web Data Commons Web Tables Corpus⁴ (WTC) 2015, which is the second large-scale extraction of web tables published during the work on this thesis. It is extracted from the July 2015 version of the Common Crawl, which contains 1.78 billion HTML pages from 15 million different web sites. This second extraction contains additional metadata and distinguishes between different types of content tables. The extraction framework is based on the one that was used for the WTC 2012 extraction and contains contributions from Eberius et al. [Eberius et al., 2015]. The additional metadata includes the table orientation, the URL and page title of the web page, the heading closest to each web table and text around the table as well as timestamps that are extracted from the web page. Such metadata can be useful for the semantic table interpretation of web tables [Yakout et al., 2012, Zhang, 2017] and the data fusion of time-dependent data [Zhang and Chakrabarti, 2013]. The following sections first describe the changes in the extraction process and the additional metadata recovery steps. Then, a data profile of the contained web tables is presented.

³<http://webdatacommons.org/hyperlinkgraph/2012-08/topology.html>

⁴<http://webdatacommons.org/webtables/#results-2015>

Extraction Process

This section describes the extraction process that was executed to create the Web Data Commons Web Tables Corpus 2015. The extraction is based on the same framework as the WTC 2012 extraction (see Section 4.4.1) and this section only describes changes and additions to this framework.

After the publication of the Web Data Commons Web Tables Corpus 2012 and the release of the extraction code as open source, Eberius et al. [Eberius et al., 2015] extended the extraction process by an advanced table classification step and contributed their changes back as open source. This made it possible to incorporate their changes in the extraction of the WTC 2015.

As for the WTC 2012, the HTML pages in the web crawl are parsed and all HTML table tags are considered as potential candidates for extraction. The table heuristic filtering step then uses the nesting heuristic and removes all tables with less than two columns or less than 3 rows, resulting in 10 billion candidate tables, on average 5.75 per web page.

Then, the classifier proposed by Eberius et al. is used to classify the tables as either `relational table`, `entity table`, `matrix table` or `layout table`. The classifier uses global features, local features and content features as described in Section 4.3.1. Applying this classifier to the 10 billion candidate tables results in 233 million content tables (2.28%), which are further classified as 139 million entity tables (1.36%), 90 million relational tables (0.88%) and 3 million matrix tables (0.03%).

On a technical level, the transformation from HTML mark-up to a tabular data structure is done in the same way as described in Section 4.4.1. However, the output format is JSON instead of CSV and includes the additional context metadata. The tabular data is stored in a two-dimensional JSON-Array in a column-oriented layout.

Metadata Recovery

In addition to earlier versions of the extractor, methods to determine the orientation of a table, i.e., horizontal or vertical, as well as methods to extract context data from the page content around the web table were added in the WTC 2015 extraction. This section describes the additional types of metadata that are provided for the web tables in the corpus as well as the methods used to generate these metadata. Figure 4.8 shows the different types of extracted metadata.

Header Row Detection. The header row detection uses a content-based heuristic. First, the values of all cells in the first three rows are transformed into content patterns. A content pattern generalises a value by replacing all characters with the identifiers of certain character classes and all subsequent occurrences of the same character class are merged. For example, the value “5.69m” is transformed to the pattern “DPDA”. The considered character classes are alphabetical (A), digit (D),

punctuation (P) and special (S) characters. After transforming the values, two rules are applied: (1) if the cells of the same column in the first and second row have the same content pattern, there is no header row; (2) else, if the cells of the same column in the second and third row have the same content pattern, then the first row is a header row.

Table Orientation. The two table types `relational` and `entity` can be further differentiated by their orientation into horizontal and vertical. In a horizontal table, the columns represent attributes and the rows represent tuples, while in a vertical table, the rows represent attributes and the columns represent tuples. The table orientation detection for relational and entity tables first checks if the header detection found a header in the first row of the table. If so, the table is horizontal. Otherwise, for all cells in a column or row, respectively, the standard deviation of the content length is calculated. If the average standard deviation of all rows is larger than the average standard deviation of all columns, then the table is horizontal, otherwise it is vertical. This is based on the intuition that all values for the same attribute should have similar content lengths and if the average deviation of the content lengths is smaller for columns than for rows, the columns likely represent the attributes.

Context Data. The additional context metadata is extracted from the URL and content of the web page containing the web table. The table heading and text before and after the table are determined from the DOM tree of the web page. The timestamps before and after the table are searched using a set of regular expressions and also determined from the `last modified` HTTP header of the original request made by the crawler.

Profiling

This section presents a data profile of the web tables contained in the extracted corpus. This profile shows the different tables types as well as the distributions of table size, column data types, and origin with respect to the top-level domain of the web pages from which they were extracted.

Table Types. Figure 4.9 shows the distribution of tables per table type among all content tables. The majority of all web tables is classified as `entity` table with 60% of all content tables, almost equally distributed between horizontal (33%) and vertical (27%). Further 38% of the web tables are classified as `relational`, with 36% being horizontal and only 2% being vertical. The type `matrix` is only assigned to 1% of the web tables. The remainder of this section will focus on the horizontal relational tables, which are the relevant web tables for the rest of this thesis.

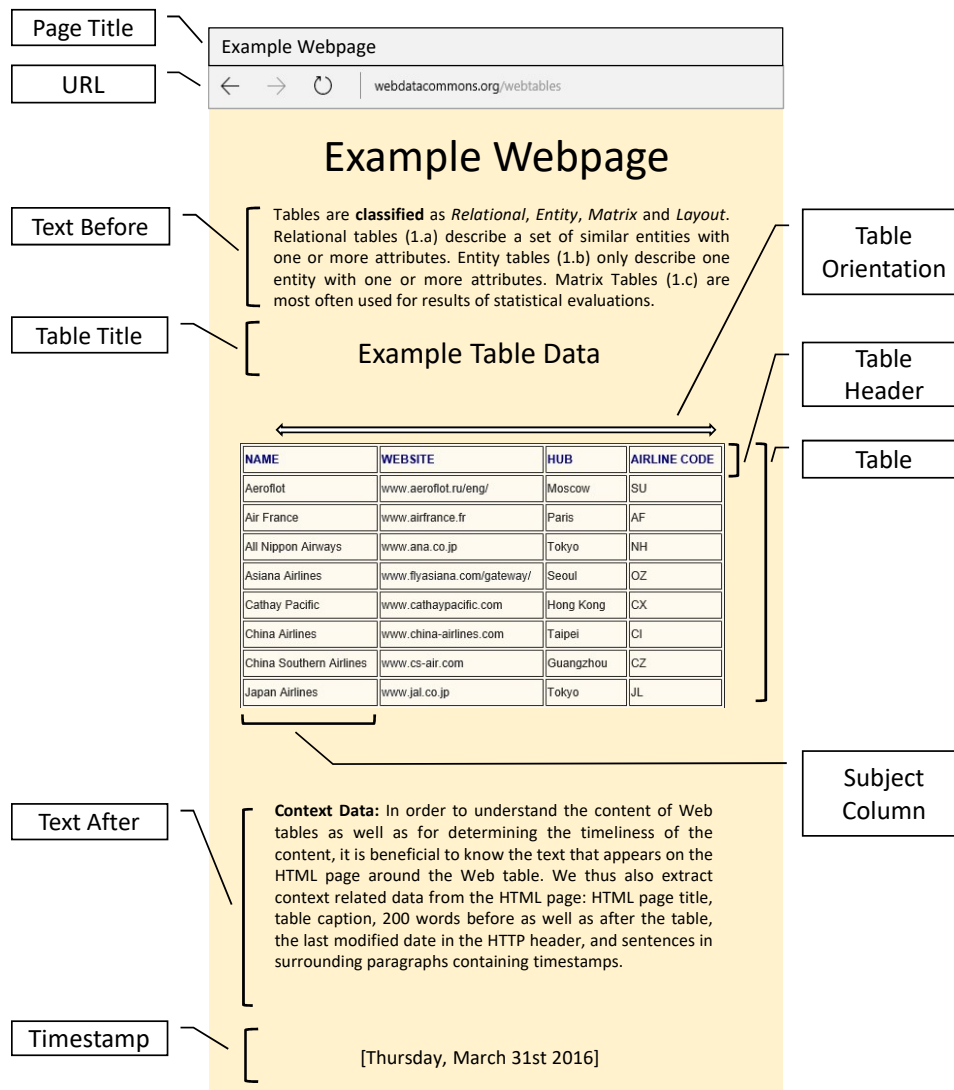


Figure 4.8: WTC 2015: Table Metadata.

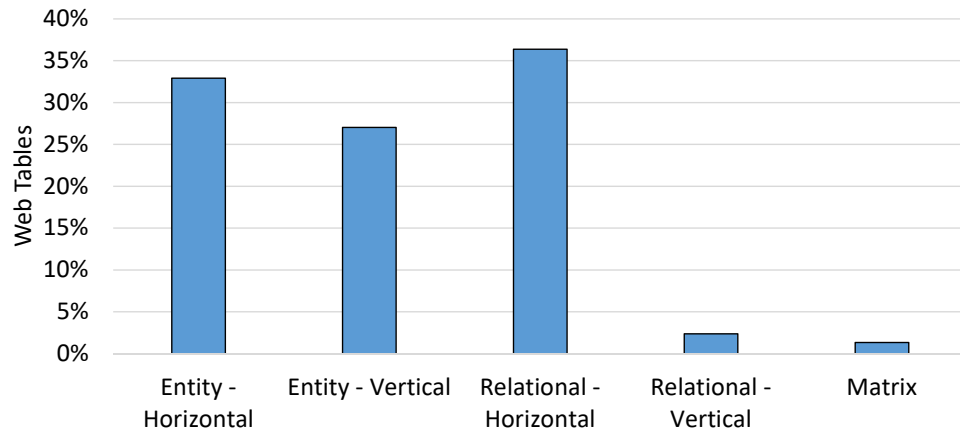


Figure 4.9: WDC Web Tables Corpus 2015: Distribution of tables over table types.

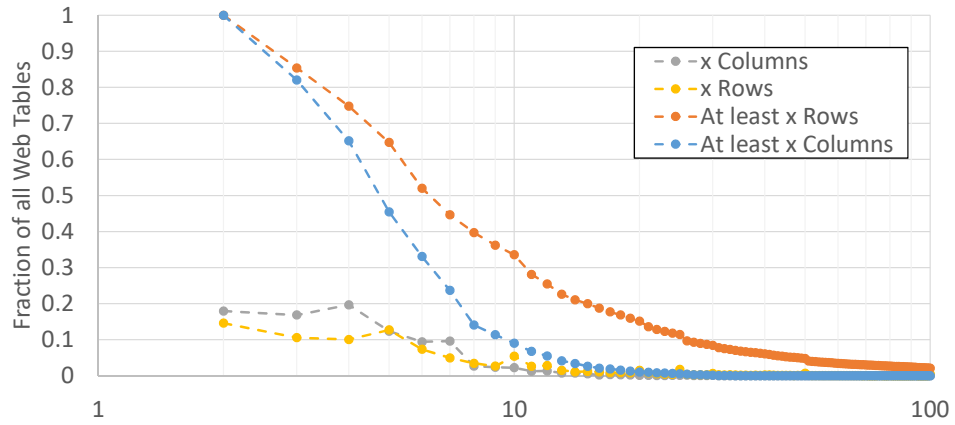


Figure 4.10: WDC Web Tables Corpus 2015: Distribution of web tables by rows and columns.

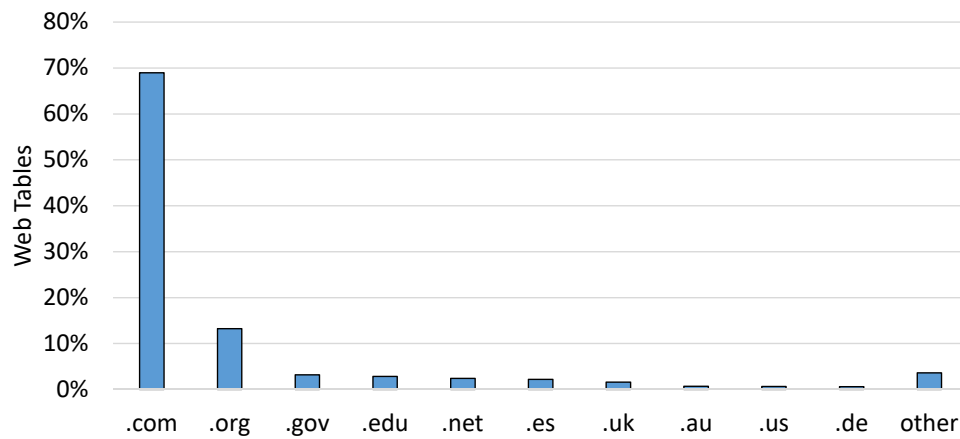


Figure 4.11: WDC Web Tables Corpus 2015: Distribution of web tables by top-level domain.

Table Size. Figure 4.10 shows the frequency distribution of web tables with respect to their size in terms of rows and columns. Similar to the WTC 2012 corpus (see Figure 4.5), the distributions show that the web tables are small. The small tables in the WTC 2015 corpus tend to contain more columns, which might be an effect of the new table type classification used for the WTC 2015 extraction. While the WTC 2012 extraction only differentiated between content and layout tables, the WTC 2015 extraction also considers other types. Especially tables of type entity table, which usually have only very few columns, are no longer considered in this statistic. As for the WTC 2012, only 25% of all tables have more than ten rows and only 2% of all tables have more than ten columns. The mode for rows remains at 2, with 15% of all tables, and increases for columns from 2 to 4, with 20% of all tables. The median is 6 for rows with an average of 14.45 and 4 for columns with an average of 5.20.

Data Types. The distribution of data types changed mostly for `numeric`, which is now the most frequent data type with 52%, followed by `string` with 47%. This large fraction of non-textual columns indicates that many web tables in this corpus contain quantitative data in addition to relations between named entities, which makes such data types increasingly relevant for semantic table interpretation methods.

Column Headers. Concerning the 50 most frequent column headers, again very general terms like “*date*”, “*name*”, or “*comments*” are found very frequently and occur in 10% of all columns in the corpus are found. The next-largest group changed from shopping for the WTC 2012 to sports for the WTC 2015 with headers such as “*team*”, “*opponent*”, or “*pts*” (points) and amounts almost 8% of all columns in the corpus. Same as for the WTC 2012, the most frequent column header is the empty string for 7% of all columns.

Table Origin. Figure 4.7 shows the distribution of web tables over the ten most frequent TLDs. The majority of all web tables still originates from web pages with the `.com` TLD, which increased from 52% to 69% compared to the WTC 2012 extraction. The distribution over the remaining TLDs, however, changed and the second-most frequent TLD is now `.org` with 13%, followed by `.gov` with 3%.

Context Metadata. In addition to the table data, this corpus also includes contextual metadata. This metadata contains the URL, the page title, the table title, 200 words from the text before and after the table as well as timestamps, if available, for each web table. Overall, 48% of all web tables have a timestamp that occurs after the table in the document, 15% have a timestamp occurring before the table, and for 21% of the web tables the last modified date of the web page is available.

4.5 Conclusion

This chapter introduced web tables and presented the various steps that are necessary for the large-scale extraction of web tables from web pages. Section 4.2 gave an overview of related work on web tables and presented common use cases. Section 4.3 then introduced the web table extraction process and its individual steps: table classification, header row detection, data type detection, and subject column detection and discussed the relevant related work for each step. Then, Section 4.4 gave an overview of existing web table corpora and presented the two web table corpora that were created during the work on this thesis.

This chapter made the following contributions:

- **Literature Survey:** This chapter introduced the web table extraction process and surveyed existing literature along the individual steps of this process. The comparison of the various approaches for each step showed that the lack of publicly available datasets hinders the comparability.
- **Web Table Corpus Profiling:** This chapter gave an overview of the existing web table corpora that have been used in the related work. This showed that only very limited information is available and the content and structure of the contained web tables is unclear. This situation was improved by the profiling and publication of two large-scale web table corpora, which shows that most web tables are small, with a median of only 6 rows in both corpora, and that 35% - 52% of the columns in web tables contain non-textual data. These are important characteristics for semantic table interpretation methods and have not been addressed sufficiently by existing work.

The web table extraction process involves many individual steps that need to be considered for the creation of a corpus of web tables. First, web tables have to be classified into a taxonomy of table types, which primarily distinguishes between layout and content tables. Content tables can be further categorised into relational tables, entity tables or matrix tables. The classification into these types is usually performed using a supervised machine learning model that uses various types of features which are derived from the content and structure of the web tables. For content tables, it is further important to recover the table schema, which includes the detection of header rows and data types for the columns in the web tables. Finally, the detection of a subject column is important for the semantic interpretation of web tables and often relies on heuristics based on the data type and uniqueness of the columns.

Although methods for the recognition and structural analysis of web tables have been published and improved for about two decades at the time of writing, the Web Data Commons Web Tables Corpus 2012 was the first large-scale, publicly available corpus of web tables that enables any interested researcher to work with web tables. This removes a severe entry barrier for research in this area, as web crawling and the processing of such crawls for the extraction of web tables

requires a large computing infrastructure and the implementation of various different methods. The data profiling of the two extracted corpora further shows two important properties of web tables, which are rarely addressed by related work: web tables are very small, with a median of only six rows in both corpora and 35% - 52% of all columns have a data type other than string. These two properties have implications for the semantic interpretation of web tables and their possible applications. This will be discussed further in Chapters 5 through 7, where it will be shown that non-string columns are a large fraction of the interpretable columns with respect to the DBpedia knowledge base and that small web tables pose severe challenges for semantic table interpretation algorithms that otherwise work well on larger tables.

Part II

Matching Web Tables to a Knowledge Base

Chapter 5

Semantic Table Interpretation

5.1 Introduction

The structural analysis of web tables, as introduced in the previous chapter, recovers the metadata of web tables and yields data sources that can be used by human consumers for tasks such as data analysis or data mining. However, to facilitate the discovery of useful web tables or their automatic processing for tasks such as knowledge base augmentation or question answering, it is also necessary to have a semantic interpretation of the web tables. A semantic table interpretation is achieved by aligning the schema of the web table with a predefined vocabulary or ontology, which defines classes and properties of real-world entities, and to link the entities described in the web table to instances of these classes. The classes, entities, and their properties are available in knowledge bases, which are hence the preferred type of reference knowledge for semantic table interpretation approaches.

This chapter introduces the semantic table interpretation tasks and surveys the approaches that have been proposed in the literature. The survey gives an overview of the used features and methods, summarises the individual approaches, and then discusses them with respect to comparability and real-world applicability. Based on the findings of this survey, which shows that both comparability and applicability are limited, a new evaluation dataset for the semantic table interpretation tasks is presented. This dataset, the T2D gold standard, contains annotations for all tasks that are necessary to generate triples for knowledge base augmentation from a web table and is made publicly available. Further, a new algorithm for semantic table interpretation, T2K Match, is proposed. In addition to named-entity columns, which are in the focus of most other approaches, this algorithm further annotates literal columns, which make up an estimated 35-52% of all columns in the Web Data Commons Web Tables corpora. To ensure reproducibility, the algorithm is published as open source.

Given a target knowledge base, semantic table interpretation means to annotate a web table with the classes, entities, and properties from this knowledge base. It is hence the process of recognising the elements that are already known, which

in turn allows the interpretation of the elements which are unknown through the structure of the web tables. Recognising the known elements is challenging, because different sources usually do not agree on common identifiers. This shows in differently spelled names, uses of homonyms or synonyms, or different modelling of schemata. Handling these challenges requires the use of schema and data matching methods, which map the elements of a web table to the elements of the target knowledge base. However, other than in a traditional data integration scenario, where a few large data sources need to be integrated, a corpus of web tables constitutes a very large number of small and heterogeneous data sources.

The method that is presented in this chapter deals with these circumstances by solving three semantic table interpretation tasks: (1) annotating a web table with a class, (2) annotating rows in a web table with entities, and (3) annotating columns in a web table with properties from the knowledge base. By iteratively applying data and schema matching methods, the results of early stages are successively refined and provide valuable information on how to execute later stages. Specifically, the first iteration determines candidate entities, which are used to find class candidates in the knowledge base. The second iteration refines these candidate entities and classes based on a preliminary property annotations. Finally, all following iterations refine the entity and property candidates until a final solution is achieved.

The contributions of this chapter are the following:

- **Literature Review:** A survey of literature on semantic table interpretation organises the commonly used features and methods according to its different tasks. The published work on the topic is further summarised and discussed with respect to comparability and real-world applicability. Current approaches are often limited with respect to these two criteria, showing the need for public evaluation datasets and more agreement on the combination of tasks which need to be solved.
- **T2D Gold Standard:** A publicly available gold standard containing web tables and annotations for the tasks of annotating web tables with classes, columns with properties, and rows with entities using DBpedia as target knowledge base is presented. In addition to earlier gold standards, T2D also contains negative examples and annotations for literal columns. This new gold standard hence allows for a more comprehensive evaluation that gives indication to the practical applicability of an algorithm.
- **T2K Match Algorithm:** A novel algorithm for semantic table interpretation is proposed that solves the tasks of annotating web tables with classes, columns with properties, and rows with entities. The schema and data matching steps are executed and refined iteratively and produce a high-quality result on the T2D gold standard, which is still comparable to newer methods that have been proposed after the original publication of the algorithm. The annotation of literal columns and increased quality are improvements over the state of the art.

This chapter is organised as follows. First, a definition of semantic table interpretation and its tasks is given, the common methods of approaching these tasks as proposed in the literature are introduced, and comparability, reproducibility as well as fitness for large-scale applicability are discussed in Section 5.2. Then, Section 5.3 introduces and profiles T2D, a new gold standard for semantic table interpretation. In Section 5.4 presents T2K Match, a new algorithm for semantic table interpretation, and its experimental evaluation is presented in Section 5.5.

Parts of the work presented in this chapter, i.e., T2K Match and the T2D gold standard, have previously been published in [Ritze et al., 2015].

5.2 Related Work

This section surveys the related work in the area of semantic table interpretation. First, the different tasks are defined and the different types of features used by methods for these tasks are introduced. Then, the approaches proposed in the literature are summarised, and finally, their comparability and real-world applicability are discussed.

5.2.1 Semantic Table Interpretation Tasks

The process of semantic table interpretation consists of several tasks which focus on understanding different parts of the schema of a web table as well as on understanding which entities the mentions in the table cells refer to. This section defines the data model that is used to represent a web table and introduces the three semantic table interpretation tasks “*Class Annotation*”, “*Relation Annotation*”, and “*Entity Annotation*”. For each task, commonly used features are presented. Finally, approaches that collectively solve multiple of these tasks are discussed.

The goal of the class annotation task is to annotate the named entity columns of a web table with the class from the knowledge base which contains the entities that are referred to by the respective column. The goal of the relation annotation task is to annotate the columns of a web table with properties from the knowledge base, which represent the relation between the subject column and the respective column. The goal of the entity annotation task is to annotate cells of named entity columns in a web table with entities from a knowledge base.

Data Model

A corpus is a set $\mathcal{T} = \{t_1, t_2, \dots, t_{|\mathcal{T}|}\}$ of relational web tables. Each web table t_i is a relation instance, referred to by lower-case letters, of a relation $T_i = \{A_1, A_2, \dots, A_{|T_i|}\}$ (the schema of the web table), referred to by upper-case letters. Every row of a web table, except the header row, corresponds to one tuple r_k in t_i and every column in the header row corresponds to an attribute A_j in T_i . Each A_j may have a name, which is the column header in the web table, if present, and

NAME	WEBSITE	HUB	AIRLINE CODE	Schema T_i
Aeroflot	www.aeroflot.ru/eng/	Moscow	SU	Tuple r_k
Air France	www.airfrance.fr	Paris	AF	
All Nippon Airways	www.ana.co.jp	Tokyo	NH	
Asiana Airlines	www.flyasiana.com/gateway/	Seoul	OZ	
Cathay Pacific	www.cathaypacific.com	Hong Kong	CX	
China Airlines	www.china-airlines.com	Taipei	CI	
China Southern Airlines	www.cs-air.com	Guangzhou	CZ	
Japan Airlines	www.jal.co.jp	Tokyo	JL	
Subject Column A_e	Attribute A_j			

Figure 5.1: The Web Table Data Model.

has a data type that is determined through data type detection (see Section 4.3.3). The web table data model is visualised in Figure 5.1. The terms *row* and *column* are used for concrete web tables, while the terms *attribute* and *tuple* are used on a conceptual level.

It is further distinguished between named entity columns and literal columns. A named entity column contains strings which are the names of entities, while literal columns are either strings that do not refer to an entity or are of a different data type. Every relational web table has a subject column A_e , which is a named entity column. The subject column contains the names of the entities that are described in the table. All other columns are assumed to represent binary relations with the subject column, i.e., $T_i : \{A_e\} \rightarrow T_i$ where $A_e \in T_i$.

Class Annotation

The goal of the class annotation task is to find a class in the knowledge base that contains the entities that are mentioned in a specific column of a web table. Finding a class annotation for the subject column of a web table, i.e., the column that contains the main entities that are described in the web table, corresponds to finding a class annotation for the web table as a whole.

Definition 16 (Class Annotation) *Given a named entity column A_j of a web table t_i , find the class c in the knowledge base that semantically describes the entities in $t_i[A_j]$.*

Approaches that annotate columns with classes from a knowledge base usually exploit the string similarity of headers and values in the table, use the results of a data matching step, or leverage external sources. Each of these features is introduced in the following paragraphs.

String Similarity. The simplest group of features is based on string similarity. It is often employed to generate initial candidates for class annotations which are then refined by later steps. It can be further differentiated into label-based and instance-based methods: Label-based methods compare the column headers in the web table to the class labels in the knowledge base using string similarity measures [Limaye et al., 2010, Buche et al., 2013], or simply use a lookup service that is provided by the knowledge base [Wang et al., 2012]. Instance-based methods compare the column content and the labels of all instances of a class using set similarity measures [Buche et al., 2013, Fan et al., 2014, Chu et al., 2015]. For such instance-based methods, all values of a column or class are concatenated and treated as documents such that standard document similarity measures such as TF-IDF and cosine similarity can be applied.

Re-using Entity Annotations. If cells have already been annotated with entities, methods can use the classes of these entities in the knowledge base to reason about the class for a column. To generate candidate classes, the union of all entities' classes [Zhang, 2017] can be created or the entities may vote for majority class [Mulwad et al., 2013, Balakrishnan et al., 2015]. More involved approaches either measure the distance between the entities' classes and the candidate class in the class hierarchy [Limaye et al., 2010] or calculate a domain consensus, which measures the similarity of all entities in the web table to the entities of the candidate class in the knowledge base [Zhang, 2017].

Class Specificity. The classes in a knowledge base are usually organised in a hierarchy and it is desirable to annotate columns with the most specific class. Hence, some approaches calculate a class specificity, independently of the web table, and use it as a feature to select an appropriate class from the generated candidates [Limaye et al., 2010, Mulwad et al., 2013]. Alternatively, such a specificity measure can also be defined as a variation of the TF-IDF scoring, which allows to model different weights for each entity label, depending on how frequently it occurs in the knowledge base [Chu et al., 2015].

External Sources. For the generation of candidate classes, it is also possible to use external sources which associate an entity mention with an entity in the knowledge base and a score or probability. Such external sources can be created by applying patterns to a large text corpus [Venetis et al., 2011] or by analysing the link texts of hyperlinks in Wikipedia or general web pages [Bhagavatula et al., 2015]. Another possibility is to use a crowd-sourcing platform to obtain annotations for some of the columns and then reason about the influence of these columns on the classes for other columns [Fan et al., 2014].

Relation Annotation

The goal of the relation annotation task is to find a property in the knowledge base that describes the relationship between the subject column and another column in the same web table.

Definition 17 (Relation Annotation) *Given a column A_j of web table t_i with subject column A_e , find the property p in the knowledge base that semantically describes the relationship between A_e and A_j , and forms the predicate of triples generated from the two columns: $\{(r[A_e], p, r[A_j]) | \forall r \in t_i\}$.*

Approaches that annotate columns with relations from a knowledge base exploit the string similarity of headers and values in the table, use the results of a data matching step, or leverage external sources. Each of these features is introduced in the following paragraphs.

String Similarity. Label-based methods compare the table title or column headers to the property labels in the knowledge base using string similarity measures [Buche et al., 2013, Zhang, 2017]. Instance-based methods compare the values in the table or the context around the table to the triples using the property in the knowledge base, by comparing the string values to the label or URI of the entities at the subject and object position of the triple [Zhang, 2017].

Re-using Class Annotations. The class annotations of columns in the web table must match the classes of the domain and range of the property in the knowledge base [Buche et al., 2013], and can hence be used to generate candidates. This constraint can further be used to quantify the compatibility of candidate relations using probabilities which are estimated from the triples in the knowledge base [Chu et al., 2015, Limaye et al., 2010].

Re-using Entity Annotations. The most common approach is to score properties by the fraction of entity pairs in two columns which appear as subject and object in a triple in the knowledge base [Limaye et al., 2010, Mulwad et al., 2013, Munoz et al., 2013, Chu et al., 2015, Zhang, 2017]. Here, a preceding data matching step has annotated the cells in the columns with entities from the knowledge base and methods query the knowledge base for triples containing these entities.

External Sources. The generation and scoring of candidate properties can also be done using external sources. Such sources contain relations between entities which have been extracted from large text corpora and can be used to estimate probabilities for the candidate properties [Venetis et al., 2011, Sekhavat et al., 2014, Balakrishnan et al., 2015]. These sources are created by learning extraction patterns for the known properties in the knowledge base. By applying these patterns to a large text corpus, many surface forms for entities which participate in the respective relation can be collected.

Entity Annotation

The goal of the entity annotation task is to find an entity in the knowledge base that corresponds to the entity that is mentioned in a cell of a web table. The entities mentioned in the subject column of the web table are representative for the whole row containing the respective cells.

Definition 18 (Entity Annotation) *Given a named entity column A_j and row r of web table t_i , find the entity e in the knowledge base that semantically describes the entity that is mentioned in $r[A_j]$.*

Approaches that annotate cells in a web table with entities from a knowledge base exploit the string similarity of cell values in the table or external sources to generate candidate entities. Each of these features is introduced in the following paragraphs.

String Similarity. The most common method to generate candidate entities is to compare the cell value to the entity labels in the knowledge base, either by exact string matching [Sekhavat et al., 2014] or using a string similarity measure [Limaye et al., 2010, Mulwad et al., 2013, Bhagavatula et al., 2015, Balakrishnan et al., 2015, Zhang, 2017, Chu et al., 2015, Efthymiou et al., 2017]. Additionally, the similarity of the values in other cells of the same row or column and the triples using the candidate entity as subject in the knowledge base can be calculated as a contextual similarity [Zhang, 2017].

Re-using Entity Annotations. If initial entity annotations are available, they can be re-scored using a semantic relatedness measure, which compares the similarity of all entities in the same row or column using their overlap of ingoing and outgoing Wikipedia links [Bhagavatula et al., 2015].

Entity Embeddings. Recently, methods are being explored that embed the entities of a knowledge base into a dense, continuous vector space which is supposed to resemble their semantic similarity, i.e., similar entities are close to each other in this vector space. Such embeddings can be used by looking up cell values in a dictionary of surface forms for entities and then comparing the embedding vectors of different entities with, for example, cosine similarity [Efthymiou et al., 2017].

External Sources. Candidate entities can also be generated by looking up the cell values in external sources, which can be created from link texts of hyperlinks to Wikipedia pages [Bhagavatula et al., 2015] or directly following a link if the cell contains a hyperlink to a Wikipedia page [Munoz et al., 2013, Bhagavatula et al., 2015]. Popularity measures, such as PageRank, of Wikipedia pages associated with entities can also be used for scoring [Mulwad et al., 2013].

Collective Inference

While each of the introduced tasks can be solved individually, it is often beneficial to solve multiple tasks collectively, as the results of each task can be re-used to improve the results for the remaining tasks. The following introduces the most common approaches to such a collective inference: probabilistic graphical models and iterative matching.

Probabilistic Graphical Models. Collective inference for multiple tasks is often modelled using a probabilistic graphical model. Such a graphical model represents the interactions between the annotations of entities, classes, and relations as interrelated random variables. Finding the best solution then corresponds to maximising the joint probability. The structure of the graphical model reflects the following intuition: entity annotations influence the class annotation in the same column and the relation annotations including this column, and vice versa [Limaye et al., 2010, Mulwad et al., 2013]. An addition to this structure is to define that all entity annotations in the same row influence each other [Bhagavatula et al., 2015].

Iterative Matching. An alternative to the formulation as probabilistic graphical model is an iterative matching step [Zhang, 2017]. Here, the class annotations and entity annotations are updated in each iteration until convergence is reached. For such an iterative matching to have an effect, the update of annotations takes the result of other annotation steps into account, i.e., uses features of the “*Re-using X Annotations*” categories as introduced above [Limaye et al., 2010, Mulwad et al., 2013, Buche et al., 2013, Munoz et al., 2013, Chu et al., 2015, Balakrishnan et al., 2015, Bhagavatula et al., 2015, Zhang, 2017].

5.2.2 Approaches

This section summarises the methods that have been proposed in the literature for the different semantic table interpretation tasks or their combinations. The first subsection presents approaches that rely on a target knowledge base for the annotation of web tables. The second subsection then discusses approaches that use open information extraction methods to generate a larger set of possible relations and classes and try to increase the coverage compared to approaches using an existing knowledge base.

Annotation with Existing Knowledge Base

The approaches introduced in the following solve the semantic table interpretation tasks as described above and use a variety of different features and matching techniques.

[Limaye et al., 2010] Limaye et al. proposed the first work to annotate web tables with elements from a knowledge base. Their method annotates cells with entities, columns with classes, and pairs of columns with binary relations between named entities using YAGO as target knowledge base. They generate candidate annotations for classes and entities using different string similarity measures, and these annotations are then used to look up candidate relation annotations. The scores of all candidate annotations are calculated using a probabilistic graphical model, which takes additional factors such as class specificity and the compatibility of entity annotations and class annotations into account. The inference step, i.e., finding an assignment for the model that maximises the joint probability, is performed using message-passing [Koller and Friedman, 2009]. For their evaluation, Limaye et al. annotate several datasets: Two datasets based on tables from Wikipedia *Wiki_Manual* and *Wiki_Link* as well as two datasets of web tables *Web_Manual* and *Web_Relations*. It is important to note that, although these datasets are not publicly available, many authors have re-used them, often in a modified version, to evaluate their own approaches. The *Web_Manual* dataset contains 371 web tables which are manually annotated with entities, classes, and relations. Due to the low amount of relations in this dataset, the authors created the *Web_Relations* dataset, in which they only annotate relations. Although their method has several hyper-parameters, the authors choose not to split their datasets. They estimate the weights required by their model on the *Wiki_Manual* dataset and choose the measure for class-entity compatibility based on the results for both the *Wiki_Manual* and *Web_Manual* datasets. Evaluated on the *Web_Manual* dataset, their method achieves an accuracy of 81.37% for entity annotation, 43.23% for class annotation and 51.5% for relation annotation (63.64% on the *Web_Relations* dataset).

[Mulwad et al., 2013] Mulwad et al. propose a method to annotate cells with entities, columns with classes, and pairs of columns with binary relations between named entities using YAGO and DBpedia as target knowledge bases. Their method solves the same set of tasks and uses features and a probabilistic graphical model which are very similar to the approach of Limaye et al. Notable differences are in the generation of entity annotation candidates and the inference algorithm. Entity annotation candidates are generated using Wikitology, a hybrid knowledge base constructed from Wikipedia, YAGO, and DBpedia, and then ranked using a supervised ranking model. This gives their approach an advantage, as more alternative names for the entities contained in the knowledge base are available. The inference over the graphical model is performed using *Semantic Message Passing*, a variation of the message-passing algorithm, which does not rely on pre-computed probability distribution tables and is supposed to scale to larger problems. However, the authors do not compare the runtime or scalability of *Semantic Message Passing* to the otherwise very similar approach proposed by Limaye et al. They evaluate their approach on Limaye’s datasets, but create a new reference alignment

to include annotations for the DBpedia knowledge base. They do not report how the hyper-parameters of their method were determined. For the class annotation task, they report an F1-measure of 57% for the Web_Manual dataset and for the relation annotation task, they report an F1-measure of 89% for the Web_Manual dataset and 86% for the Web_Relation dataset. For the entity annotation task, they report an accuracy of 63.07% for the Web_Manual dataset. Except for the entity annotation task, these performance measures are higher than the accuracy values reported by Limaye et al., but the publication contains no discussion of the comparability of the F1-measure reported by Mulwad et al. and accuracy scores reported by Limaye et al. or the changes that were introduced during the re-annotation of the datasets.

The following methods were published at the same time as or after the original publication of the T2K Method [Ritze et al., 2015] presented in this chapter and have hence not been considered during its design.

[Bhagavatula et al., 2015] Bhagavatula et al. propose a method to annotate cells with entities using Yago as target knowledge base. Other than Limaye et al. or Mulwad et al., they generate candidate entity annotations using probabilities derived from a database of hyperlinks on the web. Every link text of a hyperlink pointing to a Wikipedia page is considered as evidence that the link text represents the entity corresponding to the respective Wikipedia page. Further features for scoring the candidates are the semantic relatedness [Hecht et al., 2012, Witten and Milne, 2008] with other candidate entities for different cells in the same row or column and contextual similarity, which measures the overlap of tokens in the same row and column for a cell with tokens that have been observed on a training set. They use a probabilistic graphical model to combine these features, where each cell is represented by a variable that is connected to all other cells in the same row or column. As approximate inference algorithm they use link-based classification [Lu and Getoor, 2003], which uses a pre-trained local classifier to update the assignment of each node based on the assignment of its connected nodes in each iteration. They evaluate their approach on Limaye’s Web_Manual dataset and report an accuracy of 89.41% which is an improvement of 8 percentage points.

[Pham et al., 2016] Pham et al. propose a method to annotate columns with classes. Their approach is to train a classifier on feature vectors consisting of different similarity measures such as string, numeric and distributional similarity measures of column headers and cell values. For training, examples are generated by creating pairwise combinations of all properties in the training dataset that indicate if the properties have the same range or not. For the classification of a column, feature vectors comparing this column to each of the properties in the training dataset are created and then evaluated by the classifier. All positive predictions are then ranked by the confidence of the classifier. They evaluate their method on the T2D

gold standard, which will be described in Section 5.3, and report a mean reciprocal rank [Craswell, 2009] of 0.773, but do not discuss how this performance is related to the F1 score reported as benchmark in the original publication of the gold standard [Ritze et al., 2015] and in Section 5.5 of this chapter.

[Ermilov and Ngomo, 2016] Ermilov and Ngomo propose a method to annotate cells with entities and pairs of columns with binary relations using DBpedia as target knowledge base. They use the AGDISTIS [Usbeck et al., 2014] system to generate candidate entity annotations for a sample of all cells in the web table. This system uses labels from the knowledge base and additional surface forms as well as a graph-based disambiguation step to link strings to entities. It is, however, unclear if and which additional surface forms are used in their experiments. Then, relation annotation candidates are generated by looking up the column headers in an index that contains the labels and descriptions of properties in the knowledge base and by querying the knowledge base for pairs of values from different columns. The hyper-parameters of their method, a similarity threshold for index lookups and a classification model to determine the subject column, are tuned based on the results on all evaluation datasets. The evaluation of the method is performed on a re-sampled and re-annotated subset of the T2D gold standard and an artificially generated dataset. On the variation of T2D, their method achieves 72% recall and 39% precision, compared to 36% recall and 48% precision achieved by T2K Match. The authors do not publish their version of the T2D gold standard, so the experiment cannot be repeated for an error analysis. On the synthetic dataset, their method achieves an F1-measure of 85% while T2K Match does not produce a result. This experiment, too, is not reproducible for an error analysis, because the used dataset is no longer available online.

[Zhang, 2017] Zhang proposes a method to annotate cells with entities, columns with classes, and pairs of columns with binary relations. The approach distinguishes between a *learning* phase, which only considers a sample of the table, and an *update* phase, which iteratively refines the annotations, similar to T2K Match as described in Section 5.4. The learning phases creates candidate annotations for entities and classes, which are then re-scored in the update phase. After the iterative refinement ends, these annotations are used to list possible relation annotations by querying the knowledge base. The most notable differences to T2K Match are that the method only uses a sample of the data in each web tables, uses different features and measures for similarity calculation, and iterates until a convergence criterion is met. Zhang evaluates his method on a re-built version of Limaye’s dataset and reports an F1-measure of 82.3% for the entity annotation task, 64% for the class annotation task, and 66.2% for the relation annotation task. In these experiments, the proposed method outperforms re-implementations of the methods proposed by Limaye et al. and Mulwad et al.

[Efthymiou et al., 2017] Efthymiou et al. evaluate several methods for annotating cells with entities from a knowledge base. The methods differ in the way entity candidates are generated. The lookup-based method uses an index created from the `rdfs:label` and `rdfs:description` properties of all entities in the knowledge base. The returned candidates are constrained in a similar way as in T2K Match by using candidate class and candidate property mappings. The entity embedding method also uses index lookups, but calculates the similarity scores based on embedding vectors [Mikolov et al., 2013] of the returned candidate entities and applies a graph-based disambiguation step [Zwicklbauer et al., 2016]. They evaluate their method on the T2D gold standard and the re-creation of the Limaye dataset by Bhagavatula. Using a hybrid approach consisting of the lookup and entity embeddings methods, they achieve an F1-measure of 85% on the T2D gold standard, which is an improvement of 3 percentage points over T2K Match, and 82% on the re-created Limaye dataset. The results on the re-created Limaye dataset are not comparable to the results published by Bhagavatula, as the approach of Efthymiou et al. only annotates entities in the subject column of each web table, while Bhagavatula et al. annotate entities in all named entity columns.

[Ritze and Bizer, 2017] Ritze and Bizer present a feature utility study that analyses the impact of a large number of features on all three semantic table interpretation tasks. For their study, they extend the T2K Match method and the T2D gold standard, which are presented in this chapter. For the annotation of columns with classes, they find that a majority voting of the types assigned to entity annotations and class specificity are the strongest signals. For the annotation of cells with entities, the strongest signals are the string similarity of the cell value and the entity’s label as well as the similarity of the values of additional columns with the entity’s properties. For the annotation of pairs of columns with binary relations, the best performing features are label-based similarity and instance-based similarity using duplicate-based schema matching.

The approaches presented above differ in the way initial candidates are generated, in the specific similarity measures that are used, and in the way how interactions between entity, relation, and class annotations are modelled. While many of the approaches use the dataset created by Limaye et al., different performance measures and the re-annotation of the datasets raise doubts about the comparability of the results. Another issue is that the actual impact of the different methods for candidate generation, similarity calculation, and collective inference is not evaluated individually, so approaches can only be compared as a whole. The first issue is approached by the T2D gold standard, which will be presented in Section 5.3 and is already being used by other researchers to compare their methods. The second issue was addressed and Ritze and Bizer [Ritze and Bizer, 2017] who performed a feature utility study for the different semantic table interpretation tasks.

Annotation based on Open Information Extraction

The methods introduced in the following annotate web tables with concepts and relations obtained through open information extraction methods rather than an existing knowledge base. This approach promises a larger coverage, as the databases created by such methods are usually larger than knowledge bases, but also contain more noise.

[Venetis et al., 2011] Venetis et al. propose to annotate web tables with classes and relations from a large database that is created using open information extraction methods. Their method annotates columns with classes and pairs of columns with binary relations between, using two databases extracted from a large text corpus as target knowledge base. The authors argue that the annotation with elements from these databases increases the coverage of their approach compared to using one of the existing knowledge bases. They report that they can annotate 1.5 million of their 12.3 million web tables with their approach, while only 185 thousand and 577 thousand can be annotated with YAGO and Freebase, respectively. The first database, called *isA*-database, uses patterns to extract combinations of class names and entity names from web pages and search engine query logs, for example the text “*cities such as Berlin*” leads to the extraction “*Berlin isA City*”. The second database, called *relations*-database, uses the TextRunner extraction system [Yates et al., 2007] to extract triples for binary relations, for example “*Berlin capital Germany*”, from web pages. To annotate a table, the values of named entity columns are looked up in both databases to generate candidate annotations. Then, a maximum-likelihood model is used to estimate the probabilities of encountering the values in the table given each candidate (class or relation) annotation. The authors evaluate their method on the datasets proposed by Limaye and claim to outperform their results, however, they compare their F1 for the top-10 annotations to Limaye’s accuracy for the best annotation.

[Wang et al., 2012] Wang et al. propose to annotate web tables with classes from the Probase taxonomy. Before annotating the web tables, their method collects additional relations from web pages and query logs, similar to the approach of Venetis et al. Their method queries the Probase knowledge-API to obtain candidate classes for the web table based on the set of column headers and the set of entities in the subject column. For each of these sets, the knowledge-API returns a list of candidate classes with a score, and the method chooses the candidate with the highest product of header score and entity score. Their evaluation focuses on a search application using the annotated web tables and does not explicitly measure the performance of the class annotations.

[Balakrishnan et al., 2015] Balakrishnan et al. propose a method to annotate cells with entities, columns with classes and tables with binary relations using the Google Knowledge Graph as target knowledge base. Their approach builds on the

ideas presented by Venetis et al. and generates an entity index and a *isA* database from the knowledge base. Cells are then annotated using lookups in the entity index and the class annotation for a column is determined by a majority voting among the classes returned for the candidate entities from the *isA* database. To determine potential relations in the table, they look up known properties from a large-scale attribute name database (Biperpedia) [Gupta et al., 2014] in the text surrounding the table and its captions. It is unclear whether the discovered properties are used to annotated specific columns or the whole table. They do not present an evaluation of their approach.

[Cannaviccio et al., 2018] Cannaviccio et al. present a method to annotate pairs of columns with relations from a knowledge base. Instead of comparing the column contents to the names of entities in the knowledge base, they propose to learn language models for each relation in the knowledge base from sentences that connect the entities in these relations using a large text corpus. To annotate web tables, the entities in the cells are used to issue a search query to a web document search engine, and the candidate relations in the knowledge base are ranked according to how likely their language models would produce the sentences in the returned documents. The evaluation shows that the proposed method is able to assign relation annotations in 52 out of 80 cases where T2K Match fails, which indicates a better recall of the method. However, no evaluation on the T2D gold standard is performed which would provide a general comparison of the methods.

The literature discussed in this section indicates that only a rather small fraction of web tables can be interpreted with the data in current knowledge bases and that much more entity and relation labels need to be known. An approach for the large-scale collection of such labels is to use open information extraction methods. However, the related work in this area often does not clearly address how the additional classes, entities, and relations that are obtained with these methods can be integrated with the knowledge base. Rather, the mentioned approaches have search applications in mind, which use the annotations to match the web tables to user queries and do not rely on a knowledge base.

5.2.3 Comparability

This section discusses the comparability of the various approaches that have been introduced above. In general, the comparability is limited by two factors: (1) most approaches are evaluated on a dataset that was specifically created for the evaluation of the respective approach and cannot be re-used as they are not publicly available, and (2) different studies use different quality metrics, which are not comparable. These factors are discussed in more detail in the following and show that there is a clear need for publicly available datasets and a common evaluation methodology.

Datasets. Although many approaches for web table annotation have been proposed, comparability is rather limited. As there is no commonly agreed-upon benchmark, authors often create their own datasets to evaluate their proposed methods. However, only few make the implementation of their method or the used datasets publicly available, which hinders reproducibility or makes it impossible. For example, the datasets used by Limaye et al. [Limaye et al., 2010] are often used to evaluate methods, but due to different target knowledge bases, errors in the datasets, or outdatedness, several versions of these datasets have been created [Mulwad et al., 2013, Bhagavatula et al., 2015, Zhang, 2017]. As a result, only few of the reported performance values are actually comparable.

Quality Metrics. A second problem is that there does not seem to be agreement on which performance measure to use for evaluation. Some authors introduce an annotation “*no annotation*”, which is assigned to every element that is not annotated by a method and allows the use of accuracy as performance measure [Limaye et al., 2010, Mulwad et al., 2013, Fan et al., 2014, Bhagavatula et al., 2015].

In some of the presented studies, authors create multiple possible annotations in the ground truth and categorise them as either “*vital*”, “*okay*”, or “*incorrect*”. An annotation produced by their method is then scored with 1.0 for vital and with 0.5 for okay [Venetis et al., 2011, Mulwad et al., 2013, Zhang, 2017]. Partial scores may also be used for annotations that are on a different level in the class hierarchy than the correct annotation [Chu et al., 2015].

A frequently applied alternative to the accuracy measure is the use of precision, recall, and F1-measure, where precision is defined as the fraction of correct annotations to all created annotations, recall is the fraction of correct annotations to all annotations in the ground truth, and the F1-measure is the harmonic mean of both [Venetis et al., 2011, Buche et al., 2013, Fan et al., 2014, Chu et al., 2015, Zhang, 2017, Efthymiou et al., 2017].

Methods that produce a ranked list of annotations are either evaluated by taking the top-k elements into account [Venetis et al., 2011], resulting in precision@k and recall@k, using a ranked performance measure such as mean reciprocal rank [Pham et al., 2016], or by evaluating only the highest ranked annotation [Mulwad et al., 2013, Zhang, 2017].

This variety of evaluation methodologies makes the results incomparable and sometimes leads to inappropriate statements in the literature: Venetis et al. compare their F1-measure@10 performance to the results of Limaye et al. who were using accuracy@1, and Pham et al. compare their mean reciprocal rank to the F1-measure achieved by T2K Match, the method presented in this chapter. In both cases, a measure that takes multiple candidates into account is compared to a measure that only considers the first result.

5.2.4 Real World Applicability

The ultimate goal of semantic table interpretation approaches is to apply them to annotate a large corpus of web tables, which can then be used in downstream applications such as dataset search or knowledge base augmentation. This section discusses to which extent the proposed methods are applicable in such a setting with respect to a representative evaluation, runtime efficiency, and limitations concerning literal values.

Evaluation Methodology. Concerning the evaluation methodology, it is often unclear how the methods proposed in the literature will handle an unknown dataset. Reasons are that either evaluation data was also used for parameter tuning [Limaye et al., 2010], it is not stated which data was used for tuning [Mulwad et al., 2013], or no explicit evaluation of the annotation task is provided [Wang et al., 2012, Balakrishnan et al., 2015]. Further, some approaches are not evaluated for a real-world application. For example, evaluation data is filtered such that the respective method is always able to make an annotation [Venetis et al., 2011], or the method always links to an existing entity in the knowledge base [Bhagavatula et al., 2015], i.e., cannot handle unknown entities, which leads to incorrect annotations if the knowledge base does not contain all possible entities.

Efficiency. Efficiency is also hard to gauge, as only few studies measure the runtime of their approaches [Chu et al., 2015, Zhang, 2017] or apply them to a large corpus of tables. Based on the literature, only the methods proposed by Limaye et al., Venetis et al., Wang et al., and Efthymiou et al. have been applied to large-scale datasets [Limaye et al., 2010, Venetis et al., 2011, Wang et al., 2012, Efthymiou et al., 2017]. Limaye et al. apply their method to a corpus of 25 million web tables, but don't report any statistics about the results. Venetis et al. were able to annotate 1.5 million web tables out of a corpus of 12.3 million web tables with concepts from their isA database. Wang et al. report that they discovered 10 million entities, 150k attributes and 1 million relations from a corpus of 69 million tables. Efthymiou et al. evaluate their method on a dataset of 485 thousand web tables from Wikipedia.

Data Types. Finally, most of the proposed methods focus exclusively on the annotation of named entity columns, although a large fraction of the columns in large web table corpora are numeric [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011, Wang et al., 2012, Munoz et al., 2013, Sekhavat et al., 2014, Fan et al., 2014, Chu et al., 2015, Bhagavatula et al., 2015]. Only the approaches proposed by Buche et al., Pham et al., and Zhang are able to annotate numeric columns [Buche et al., 2013, Pham et al., 2016, Zhang, 2017].

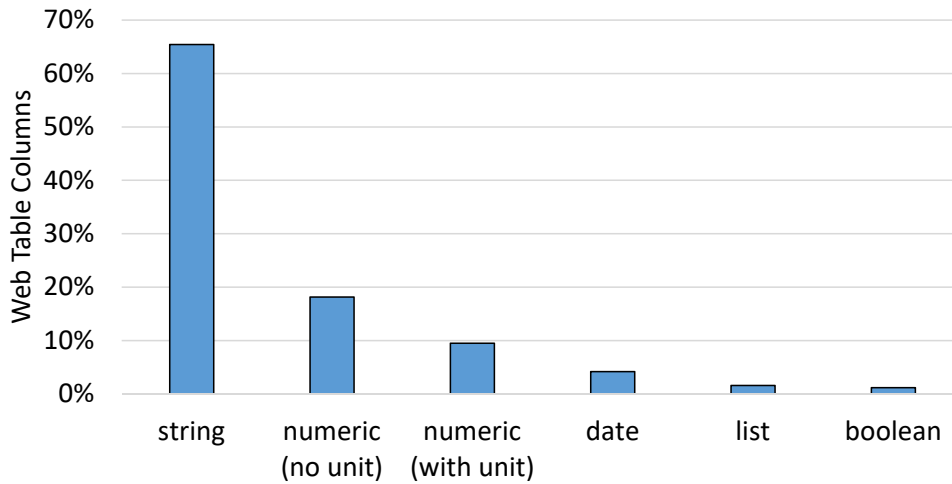


Figure 5.2: T2D: Data Type Distribution.

5.3 T2D Gold Standard

This section introduces the T2D Gold Standard.¹ The gold standard was developed to enable researchers to measure the performance of semantic table interpretation algorithms for several annotation tasks based on a publicly available dataset. The following sections describe the sampling methodology, report the data profile for each of the annotated semantic table interpretation tasks, and explain the annotation procedure that was applied for the creation of the gold standard.

5.3.1 Annotated Semantic Table Interpretation Tasks

This section presents a data profile for the T2D gold standard with respect to the annotated semantic table interpretation tasks and two different subsets that were created based on the available annotations: the schema-level gold standard with 1 748 web tables, which contains class and relation annotations, and the instance-level subset with 233 web tables, which also contains entity annotations.

The schema-level gold standard contains annotations that indicate the type of table (layout, content), and, if applicable, correspondences to a class in DBpedia for the whole table as well as correspondences to properties in DBpedia for individual columns. In total, there are 762 content tables with correspondences to 91 different classes and a total of 2 084 schema-level correspondences to 298 different properties in DBpedia.

Figure 5.2 shows that the distribution of data types in content tables is equal to the distribution of data types in the whole corpus, i.e. 65% strings and 28% numeric. The distribution of tables in the gold standard with respect to size is also designed to be close to the distribution in the Web Data Commons Web Tables

¹<http://webdatacommons.org/webtables/goldstandard.html>

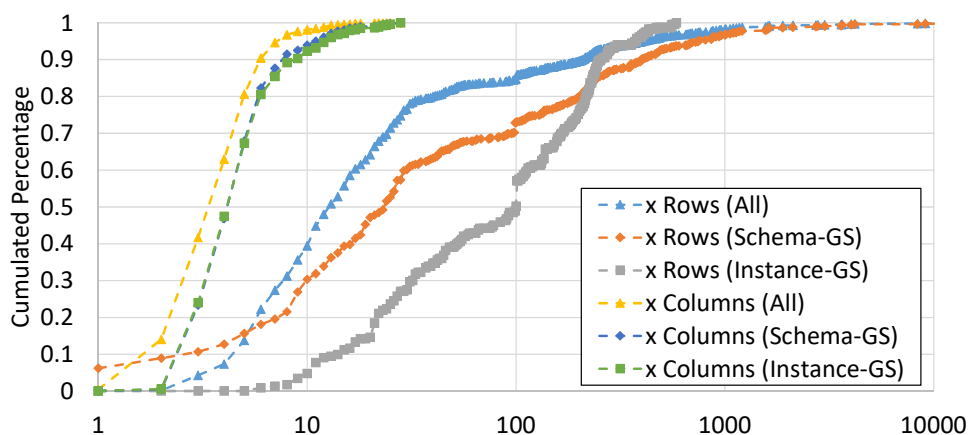


Figure 5.3: T2D: Table Size Distributions.

Corpus 2012. Small tables with ten rows or less make up 39% of the tables in the gold standard, medium-sized tables with up to 50 rows make up another 43%, and large tables with more than 50 rows make up the remaining 18%. Figure 5.3 shows the cumulative frequency of content tables by table size in terms of rows and columns.

There is, however, a bias towards larger tables, especially in the instance-level subset, which is due to two reasons. First, the goals of having a large number of entity annotations and reflecting the actual distribution of table sizes are contradicting given a limited amount of time for annotation. There is a non-negligible annotation overhead per table before an annotator can create entity annotations, which includes the annotation of the table type, data types, relations, etc., while the annotation time per entity annotation is comparably low. Hence, sampling web tables with very few rows drastically increases the annotation time compared to sampling fewer, but larger web tables with the same total amount of rows. Second, the sampling approach which is used to find web tables with entities that exist in the knowledge base ranks the web tables by their entity overlap, i.e., prefers larger tables. Completely random sampling, however, is not feasible as the amount of web tables that can be mapped to the knowledge base is very low, i.e., random sampling would require much larger sample sizes and lead to a disproportionate annotation overhead for non-matching web tables.

On the schema level, the 762 tables which can be mapped to DBpedia have a total of 4 100 columns of which 2 084 are mapped to 283 different properties in DBpedia. Of these mapped columns, 35% are the subject columns of their respective web tables. On average, web tables for the `PopulatedPlace` class have the highest number of relation annotations, with 3.3 per web table, and web tables for the `Species` class have the lowest number with only 1.7 per web table. Figure 5.4 shows the frequency distribution of the most frequent properties in the annotations. The most frequent properties are those of the `PopulatedPlace` class (language, currency, populationTotal) and the `Organization` class

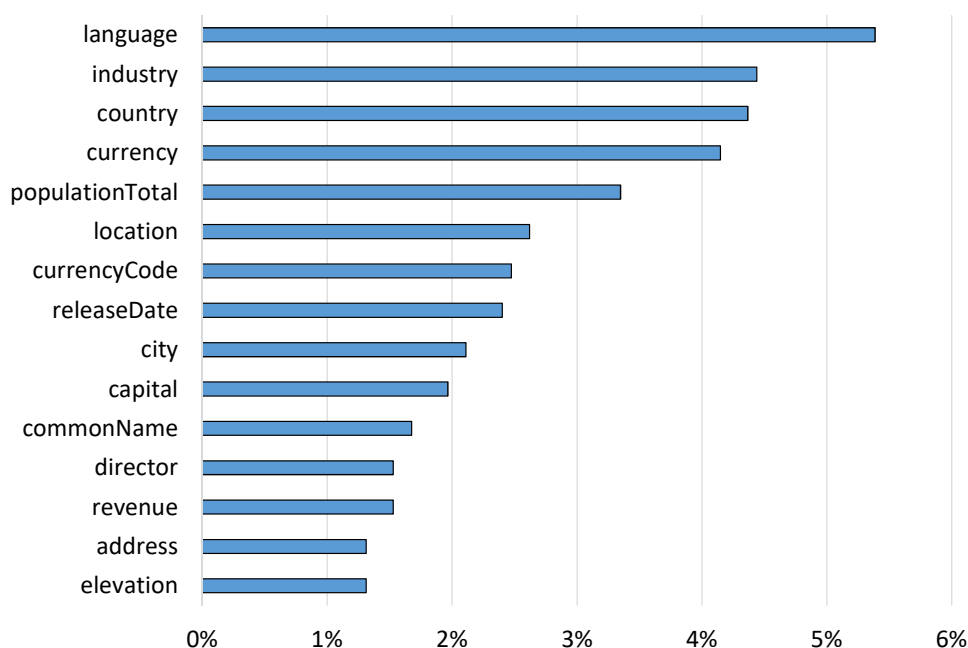


Figure 5.4: T2D: Property Frequency Distribution.

(*industry*, *country*, *location*). This is partly due to the fact that these two high-level classes have the largest amounts of web tables in the sample. However, it also indicates that these properties seem to be agreed upon as important by the creators of the different web tables.

The instance-level gold standard further contains annotations with entities in DBpedia for all rows in the web tables that have a corresponding entity. In total, there are 26 124 entity annotations and 2 442 web table rows are annotated as non-matching. On average, the *Species* class has the largest number of entity annotations with 250.1 per web table and *Organization* has the fewest with only 64.8 per web table. Figure 5.5 summarises the presented statistics for the top-level classes in the schema-level and instance-level gold standard. The most frequently annotated class is *PopulatedPlace*, which has on average rather large tables and many columns which can be mapped to properties in DBpedia. The second most frequent class is *Organization*, which has on average the smallest web tables and hence comparably few entity annotations.

More details about the characteristics of the web tables for the different high-level classes are given in Figure 5.6. The left chart in Figure 5.6 shows the average number of rows and the average number of entity annotations per table for different high-level classes. It can be seen that the *Species* class has by far the largest tables, with an average of 273 rows, and the smallest tables are found for the *Organization* class, with an average of 69 rows. The chart further shows that, for all classes, most of the rows can be annotated with entities.

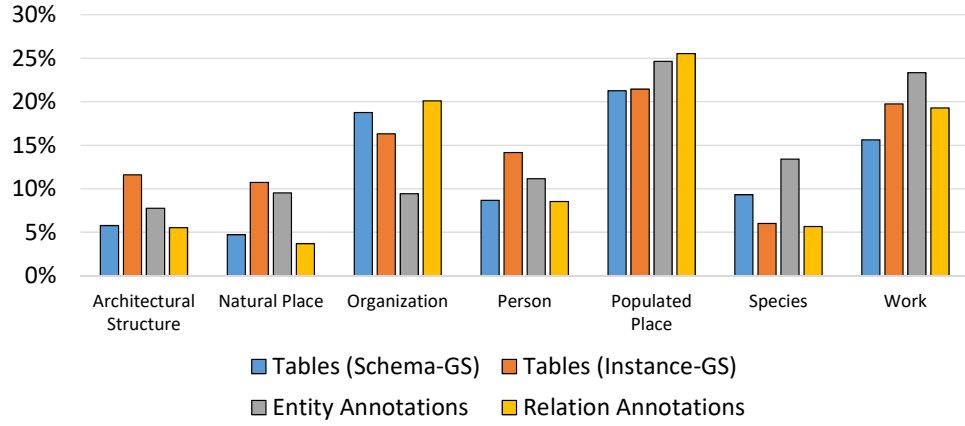


Figure 5.5: T2D: Class distribution for instance-level gold standard.

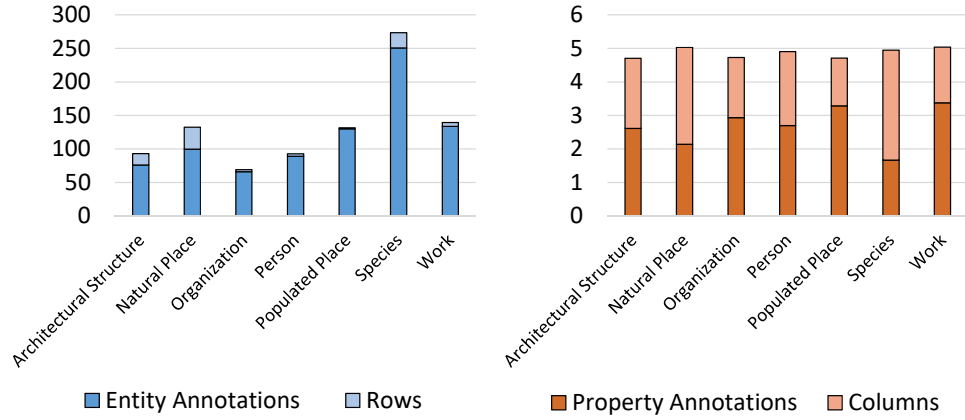


Figure 5.6: T2D: Average number of rows and columns and their annotations with entities and properties by class.

The right chart in the same figure shows the analogous statistic for columns and property annotations. The web tables for all classes have an average number of columns around 5, but the amount of property annotations varies. The largest average number of property annotations is found for the `PopulatedPlace` class, with an average of 3.3 annotated columns per web table, and the lowest is found for the `Species` class, with an average of 1.7 annotated columns per web table. These numbers include the annotation with `rdfs:label` for the subject column of each web table, so every web table that can be mapped to the knowledge base has at least one annotated column.

5.3.2 Annotation Procedure

This section describes the annotation procedure and the dedicated annotation tool that is designed for the creation of the T2D gold standard.

Table Sampling

The initial table selection is a random sample of relational web tables from the WTC 2012. The majority of these tables, however, cannot be mapped to the DBpedia knowledge base or are non-relational (i.e., errors made by the table type classifier). These web tables are included in the gold standard as *negative examples*, i.e., a semantic table interpretation method should not create any annotations for them. However, the number of *positive examples*, i.e., web tables that can be mapped to DBpedia, is too small using random sampling and would require a large sample size and annotation overhead to include a sufficient amount of positive examples. To increase the amount of positive examples, a focused sampling is used to select web tables that likely match one of the classes in the DBpedia knowledge base. This sample is generated by running the Mannheim Search Joins Engine [Lehmberg et al., 2015], a search engine for web tables, using the entities in DBpedia as query, which results in a list of web tables which contain the entities in the query and can hence be mapped to DBpedia.

Annotation Tool

The gold standard is created by two annotators with the help of a custom annotation tool that is developed specifically for this annotation task. The tool provides the annotator with a user interface, shown in Figure 5.7, that shows the original web page, highlights the web table that is being annotated and shows the extracted table data in a separate area. The annotator is then guided through an annotation process with several steps that create different types of annotation for the table. Note that the each annotator works on a distinct set of web tables to enable a larger size of the gold standard and thus, no inter-annotator agreement can be calculated.

The first step in the annotation process is to select the table type. Possible options are:

- **Broken:** The table was incorrectly extracted and cannot be used.
- **Layout:** The table is used for layout purposes and does not contain relational data.
- **Content:** The table is relational and contains data.
- **Complex:** The table has a complex layout, i.e., multi-dimensional header rows or headers in both rows and columns.
- **Useless:** The annotator cannot understand the purpose of the table.

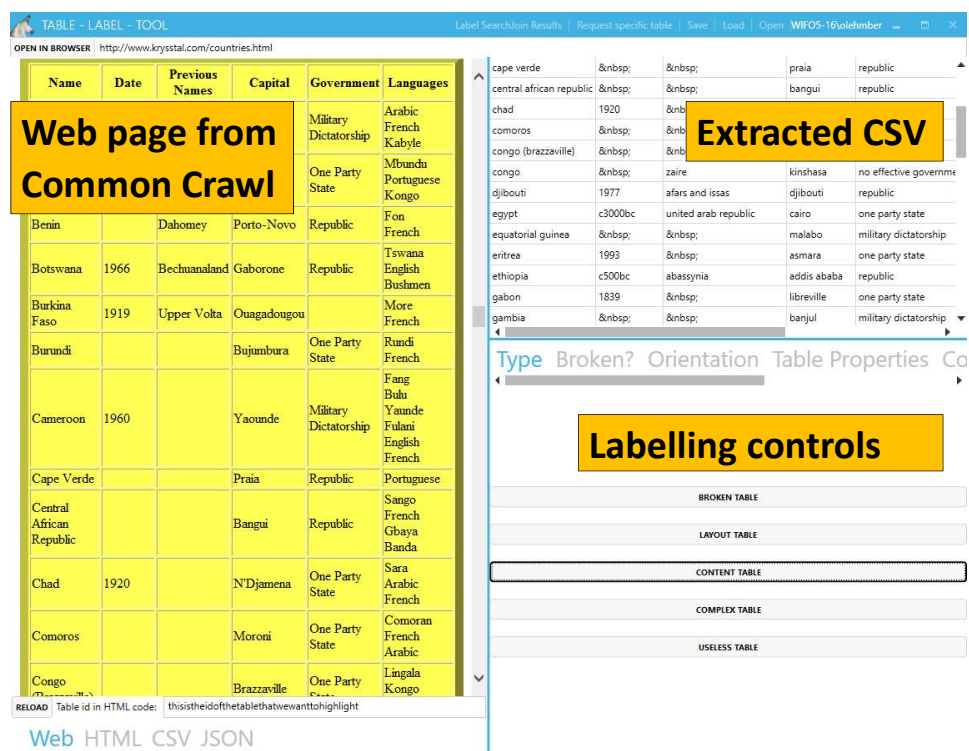


Figure 5.7: T2D: User interface of the annotation tool.

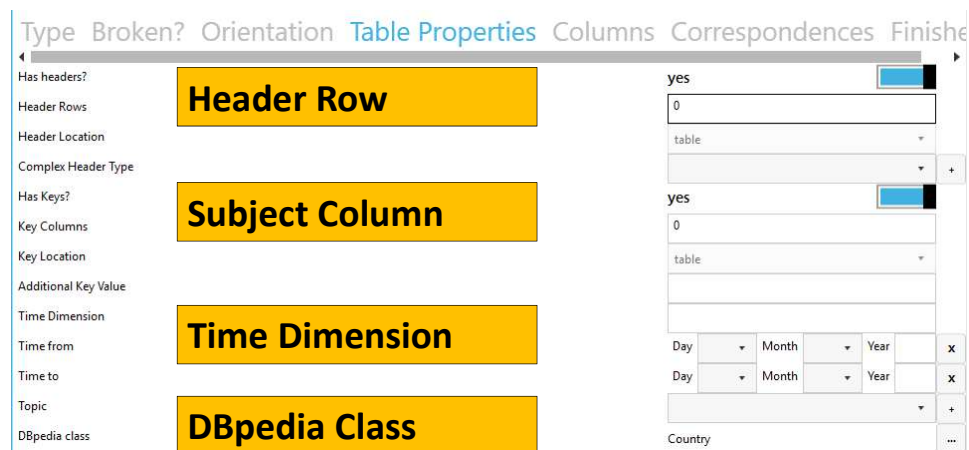


Figure 5.8: T2D: User interface for table-level metadata.

The interface displays a table with the following columns: name, date, previous names, capital, government, and languages. The 'capital' column is highlighted. To the right of the table, there are several input fields and dropdown menus for configuring the 'capital' column's metadata. These include:

- Is a useful column?**: A dropdown menu set to 'yes'.
- Column header?**: A text input field containing 'capital'.
- Data Type**: A dropdown menu set to 'string'.
- Unit Name**: An empty text input field.
- Sub Unit**: An empty text input field.
- Time Dimension**: A dropdown menu set to 'Time Dimension'.
- Time Dimension (exact)**: A set of dropdown menus for 'Day', 'Month', and 'Year'.
- DBpedia Property**: A dropdown menu set to 'capital'.

Figure 5.9: T2D: User interface for column-level metadata.

The interface displays a table with the following columns: name, date, previous names, capital, government, and languages. The 'capital' column is highlighted. Below the table, there is a 'DBpedia Lookup' button and a list of search results for 'angola'. The search results include:

- Angola**: <http://dbpedia.org/resource/Angola>. Angola, officially the Republic of Angola, is a country in southern Africa bordered by Namibia on the south, the Democratic Republic of the Congo on the north, and Zambia on the east.
- Portuguese Angola**: http://dbpedia.org/resource/Portuguese_Angola. Angola (also Portuguese West Africa, Portuguese Angola; since 1951 Overseas Province of Angola and finally State of Angola, since 1972) is the common name by which the Portuguese referred to the territory of present-day Angola.
- People's Republic of Angola**: http://dbpedia.org/resource/People's_Republic_of_Angola. The People's Republic of Angola was a self-declared socialist state (commonly known as a "communist state" in the West) that was established in 1975 after it was granted independence from Portugal.
- Dutch Loango-Angola**: http://dbpedia.org/resource/Dutch_Loango-Angola. Loango-Angola is the name for the possessions of the Dutch West India Company in contemporary Angola and the Republic of the Congo. Notably, the name refers to the colony of the Loango-Angola.
- Luanda**: <http://dbpedia.org/resource/Luanda>. Luanda, formerly named São Paulo da Assunção de Loanda, is the capital and largest city of Angola. Located on Angola's coast with the Atlantic Ocean, Luanda is both Angola's chief port and its largest city.
- U.S. Route 20**: http://dbpedia.org/resource/U.S._Route_20. U.S. Route 20 (US 20) is an east–west United States highway. As the "0" in its route number implies, US 20 is a coast-to-coast route. Spanning 3,365 miles (5,415 km), it is the longest road in the United States.
- UNITA**: <http://dbpedia.org/resource/UNITA>. The National Union for the Total Independence of Angola (Portuguese: União Nacional para a Independência Total de Angola) (UNITA) is the second-largest political party in Angola.
- Angolan Civil War**: http://dbpedia.org/resource/Angolan_Civil_War. The Angolan Civil War was a major civil conflict in the Southern African state of Angola, beginning in 1975 and continuing, with some interludes, until 2002. The war began immediately after the country gained independence from Portugal.

Figure 5.10: T2D: User interface for row-level metadata.

If the annotator chooses *Content*, the annotation proceeds with the next step. In all other cases the annotation process ends and the next table is shown. For content tables, the next step is to specify the orientation of the table. Horizontal tables represent attributes as columns and records in rows, while vertical tables represent attributes in rows and records in columns. If the annotator selects *Vertical*, the table is transposed.

The next step is the annotation of table-level metadata. The annotator specifies if the web table has one or more header row(s) or a subject column and specifies their location. She can further enter a time dimension that might be stated on the web page, such as “*last updated on*” and select a class from DBpedia for the content of the web table, as shown in Figure 5.8.

Then, the next step is the annotation of column-level metadata. The annotator specifies for each column whether it contains any data, its data type including units of measurement for numeric data, a potential time dimension, and a property from DBpedia, as shown in Figure 5.9.

The final step is the annotation of row-level metadata. The annotator selects a matching entity from DBpedia for each row in the table or none, if there is no matching entity, as shown in Figure 5.10. This selection uses the DBpedia lookup service to find candidate entities based on the cell value of the subject column for each row. For each lookup result, the `rdfs:label`, URI, and abstract are shown to the annotator. If none of the candidates match, the annotator performs a manual search in DBpedia for a matching entity and can enter its URI.

5.4 Method: T2K Match

This section describes T2K Match², a matching algorithm for iterative data and schema matching. In the following, first the pre-processing steps are explained, and then the individual steps of the matcher workflow are defined. The pre-processing steps normalise the values in the web tables and enrich the knowledge base with additional surface forms to deal with the high level of heterogeneity in web tables. The matching steps start with a candidate selection for entity and class annotations. Then, for each of these candidates, similarity values are calculated, which are used in the following steps to determine relation annotations. Finally, an iterative matching step refines the relation and entity annotations.

5.4.1 Pre-processing

This section describes the pre-processing steps that are applied to the web tables and the knowledge base before the execution of the matching algorithm. These steps perform several data cleaning and normalisation operations and add additional entity names to the knowledge base.

Data Cleaning. Several data cleaning and normalisation steps are performed on the web tables before the execution of the matcher. The first step removes HTML artefacts, special characters and additional whitespaces. Then, all string values are transformed into lower-case and value lists are split into individual values based on a regular expression. Further, a set of hand-crafted transformation rules is used to resolve abbreviations, for example “*co.*” is transformed into “*company*”. The second step performs a data type and unit of measurement detection for each column in the web table (see Section 4.3.3). All numeric values are parsed and if a unit of measurement was detected, the corresponding values are converted into a base unit. For example, $8mi^2$ is converted to 20.72 million m^2 . The last step detects a subject column using the following heuristic: The subject column is the most-unique, string-type column with an average length of at least four characters. In case of a tie, the left-most candidate is chosen (see Section 4.3.4).

²<https://github.com/olehmborg/T2KMatch>

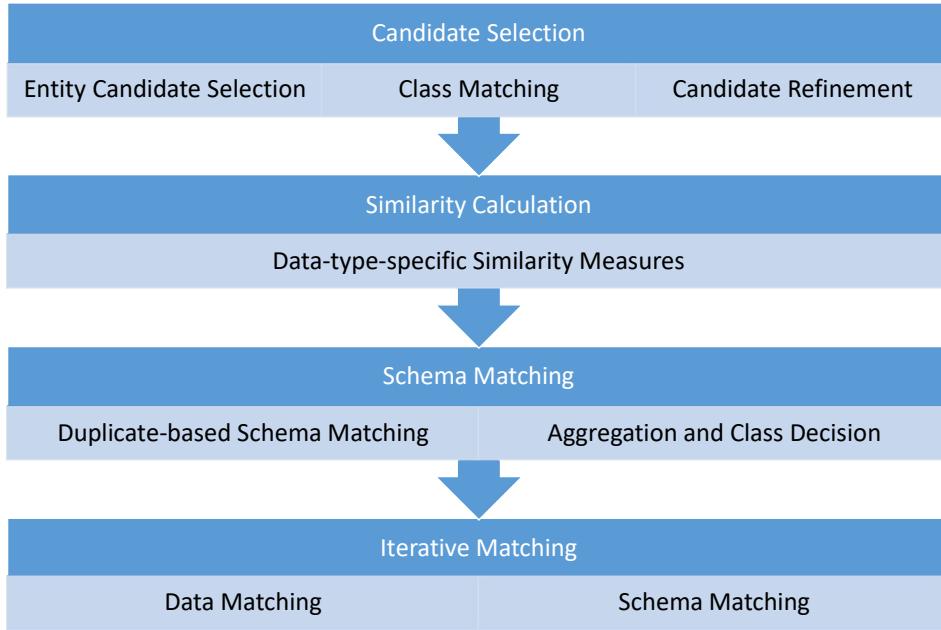


Figure 5.11: T2K Match Workflow.

Surface Form Handling. The values in a web table might use different surface forms for the names of the entities in DBpedia than the ones that are provided in the `rdfs:label` property. To be able to recognise alternative names, a surface form catalogue [Bryl et al., 2015] used to add additional entity labels to the knowledge base. The used surface form catalogue is created from anchor-texts of intra-Wikipedia links, Wikipedia article titles and disambiguation pages. In addition to this catalogue, the labels of all pages that redirects to a page corresponding to an entity in Wikipedia are added as possible labels for the respective entity.

Kurtosis Filter. Web tables often have attributes that contain row numbers or represent positions in rankings. Such attributes cannot be mapped to DBpedia as they are only meaningful in the context of the respective web tables. A characteristic of these columns is that each distinct value in a ranking column appears only once, resulting a uniform distribution. One feature of the uniform distribution that distinguishes it from other distributions is its Kurtosis value of -1.2 , as defined in Equation 5.1, which allows for the detection and exclusion of such columns from the matching process.

$$\text{Kurt}[X] = \frac{\mathbb{E}[(X - \mu)^4]}{(\mathbb{E}[(X - \mu)^2])^2} \quad (5.1)$$

5.4.2 Matcher Workflow

This section describes the steps of the T2K Match algorithm, as depicted in Figure 5.11. First, candidate entities from DBpedia are determined for each value in the subject column by index lookups on the `rdfs:label` values of all entities in the knowledge base. These candidate entities are then used to create an initial schema mapping using a duplicate-based schema matcher, and to find candidate classes for the web table. Based on the candidate classes, new candidate entities are selected. Then, an iterative phase of the algorithm alternates between re-scoring the schema correspondences and entities correspondences. In the following, the steps of the algorithm are described in more detail.

Running Example. To exemplify the steps of the algorithm, the web table shown in Figure 5.12 will serve as a running example in the following sections. The table lists five movies with their title, release year, and director as well as two ranking attributes. The correct mapping should assign the `dbo:Film` class to the whole table and the properties `rdfs:label` to the “*Title*” column, the `dbo:releaseDate` property to the “*Year*” column, and the `dbo:director` property to the “*Director*” column. The remaining two columns have no corresponding properties in the DBpedia ontology. The first column “*Fans’ Rank*” is filtered out by the kurtosis filter (it has a kurtosis of -1.2). Further, all rows except the third row (“*Manhattan*”) have corresponding entities to which they should be mapped.

	Fans' Rank	Title	Year	Director	Overall Rank
R1	1	Groundhog Day	1993	Harold Ramis	130
R2	2	Duck Soup	1933	Leo McCarey	56
R3	3	Manhattan	1979	Woody Allen	135
R4	4	Crouching Tiger, Hidden Dragon	2000	Ang Lee	165
R5	5	Requiem for a Dream	2000	D. Aronofsky	434

Figure 5.12: Running Example: Example Table.

1. Candidate Selection

The candidate selection step creates correspondences to candidate entities for each row and candidate classes for the whole web table. This initial selection limits the computational cost of later steps and corresponds to a blocking step by indexing. To obtain a set of candidate entities for the rows in a web table, an index lookup for the value of the subject column is performed. The index is created from all `rdfs:label` values of the entities in the knowledge base. All retrieved candidates are ranked by a similarity function and the top- k candidates are kept. Both the similarity function and the value of k are parameters of the algorithm. Given the web table's rows Row and entity candidates E , this step results in the similarity matrix S_{cand} , defined in Equation 5.2.

$$S_{cand} : Row \times E \rightarrow [0, 1] \quad (5.2)$$

Candidate classes are generated from the highest ranked candidate of each row in S_{cand} , referred to as \hat{S}_{cand} , i.e., all other scores in \hat{S}_{cand} are set to zero. Given the entities E and the classes C in the knowledge base, the matrix E_{class} defines the assignment of entities to classes as shown in Equation 5.3.

$$E_{class} : E \times C \rightarrow \begin{cases} 1 & \text{entity belongs to class} \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

The class similarity S_{class} shown in Equation 5.4, where $\lceil \cdot \rceil$ indicates the ceiling function, scores each class c with its relative frequency among the candidates in \hat{S}_{cand} , i.e., if 50% of the candidates belong to class `Country`, it is scored with 0.5.

$$S_{class}(c) = \frac{1}{|Row|} \sum_{r \in Row, e \in E} \lceil \hat{S}_{cand}(r, e) \cdot E_{class}(e, c) \rceil \quad (5.4)$$

All candidate classes above a configurable threshold are kept as candidates. If none of the classes exceeds the threshold, the top- k ranked classes are chosen. The selected classes are then used to refine the candidate entities through a second index lookup. This time, however, the search is limited to entities that belong to one of the selected classes. Again, candidates are ranked by a similarity measure and the top- k are kept. The similarity measure and k for the refinement step can be chosen independently of the parameters for the initial candidate search. The result of this step updates the similarity matrix S_{cand} .

For the running example, the candidates shown in Figure 5.13 are generated and the similarity scores are calculated with Jaccard similarity on word tokens. Based on the initial class similarity scores, the classes `Organisation` and `Book` as well as the entity candidates assigned to these classes are discarded. To keep the example simple, assume that the refinement step does not add any additional entity candidates.

Row	Entity	rdfs:label	Class	S_{cand}
R1	E1	Groundhog Day (film)	Film	0.67
R1	E2	Groundhog Day	Holiday	1.0
R2	E3	Duck Soup (1933 film)	Film	0.5
R2	E4	Duck Soup (1927 film)	Film	0.5
R3	E5	Manhattan	City	1.0
R3	E6	Manhattan Project	Organisation	0.5
R4	E7	Crouching Tiger, Hidden Dragon	Film	1.0
R4	E8	Crouching Tiger, Hidden Dragon (soundtrack)	Album	0.8
R4	E9	Crouching Tiger, Hidden Dragon (novel)	Book	0.8
R5	E10	Requiem for a Dream (film)	Film	0.8
R5	E11	Requiem for a Dream	Album	1.0



Class	S_{class}
Film	0.4
Holiday	0.2
City	0.2
Organisation	0.0
Album	0.2
Book	0.0

Figure 5.13: Running Example: Generated Candidate Entities.

S_{inst}	Candidate Class	Film		Album	
	Column	Title	Director	Title	Director
Row	Entity	rdfs:label	dbo:director	rdfs:label	dbo:artist
R1 (Groundhog Day)	E1	0.67	1.00	0.00	0.00
R2 (Duck Soup)	E3	0.50	1.00	0.00	0.00
	E4	0.50	0.00	0.00	0.00
R4 (Crouching Tiger, Hidden Dragon)	E7	1.00	1.00	0.00	0.00
	E8	0.00	0.00	0.80	0.00
R5 (Requiem for a Dream)	E10	0.80	0.50	0.00	0.00
	E11	0.00	0.00	1.00	0.00

Figure 5.14: Running Example: Calculated Similarity Values.

2. Similarity Calculation

Using the set of refined candidate entities, the instance-based similarity matrix between the values in the web table and the property values of the entities in the knowledge base, S_{inst} , can be calculated. These similarity values are used to calculate the schema-level and instance-level correspondences in later steps of the algorithm. Similar to the entity-class matrix E_{class} , the assignment of properties to classes in the knowledge base is defined by P_{class} , as shown in Equation 5.5.

$$P_{class} : P \times C \rightarrow \begin{cases} 1 & \text{property's domain is } C \\ 0 & \text{otherwise} \end{cases} \quad (5.5)$$

Instance-based similarities are only calculated if the data type (range) of the property in the knowledge base matches the detected data type for the column in the web table. If the property is an object property, i.e., the objects of triples of this property are entities, the data type is defined to be string and the similarity measure is calculated between the cell's value and the `rdfs:label` of the property value. The similarity measures used in this step are parameters of the algorithm and can be configured individually for each data type. For multi-valued cells or properties, the maximum of any combination of the values is chosen.

Given the web table's rows Row , columns Col , knowledge base entities E and knowledge base properties P , the instance-based similarity S_{inst} is defined as in Equation 5.6.

$$S_{inst} : Row \times E \times Col \times P \rightarrow [0, 1] \quad (5.6)$$

In the running example, the instance-based similarity scores between the web table's columns and the properties of the remaining candidate classes `Film`, `Album`, `Holiday`, and `City` are calculated. For brevity, Figure 5.14 only shows the

scores for the “*Director*” column and the two properties `director` and `artist` for the classes `Film` and `Album`. The values in the web table exactly match the director names stated in DBpedia for three of the movies in the web table and the `director` property is assigned a score of 1.0 in these cases and a score of 0.5 for the film “*Requiem for a Dream*”, as the director’s name in the web table is abbreviated. The 1927 version of the Film “*Duck Soup*”, however, was directed by “*Fred Guiol*” and does not match the value “*Leo McCarey*” in the web table.

3. Schema Matching

The third step of the algorithm aggregates the similarity values that were calculated in the second step into schema-level correspondences between the web table and a class as well as between the columns of the web table and the properties in the knowledge base.

The similarity score between a column and a property is the sum of all instance-based similarities over the values in the column, weighted by the score of the entity candidate of the corresponding rows in the web table. This resembles a weighted voting among the entity candidates and is also known as duplicate-based schema matching (see Section 2.4). The number of entity candidates that are considered for each row in this voting can be constrained by parameters of the algorithm, either by specifying that the top- k candidates for each row or all candidates above a similarity threshold are selected. The selected candidates per row will again be referred to as \hat{S}_{cand} .

Each combination of a table row and a selected entity candidate then votes for the n property correspondences with the highest similarity per column. The selected similarity scores will be referred to as \hat{S}_{inst} . Each vote can further be constrained by a similarity threshold, i.e., if the similarity score is below the threshold θ , shown in Equation 5.7, no vote is cast. Both the number of votes per column and the similarity threshold are parameters of the algorithm.

$$t(x, \theta) = \begin{cases} x & \text{if } x \geq \theta \\ 0 & \text{else} \end{cases} \quad (5.7)$$

The weighted voting for a column and property candidate is then the sum of all similarity scores over all rows and entity candidates, divided by the total number of votes $n \cdot |\text{Row}|$. For the calculation of the similarity between the subject column of the web table and the `rdfs:label` property, however, the candidate similarity is set to 1, as it is based on the comparison of the same values as the instance similarity. This results in the property similarity S_{prop} , shown in Equation 5.8.

$$S_{\text{prop}}(c, p) = \frac{1}{n \cdot |\text{Row}|} \sum_{r \in \text{Row}} \sum_{e \in E} t(\hat{S}_{\text{inst}}(r, e, c, p) \cdot \hat{S}_{\text{cand}}(r, e), \theta) \quad (5.8)$$

The property similarity scores are then further summed up to the class level to create similarity scores for each candidate class S_{class} . The class with the highest

$\hat{S}_{inst} \cdot \hat{S}_{cand}$		Candidate Class		Album	
		Column		Title	Director
Row	Entity	rdfs:label	dbo:director	rdfs:label	dbo:artist
R1 (Groundhog Day)	E1	0.670	0.670	0.000	0.000
R2 (Duck Soup)	E3	0.500	0.500	0.000	0.000
R4 (Crouching Tiger, Hidden Dragon)	E7	1.000	1.000	0.000	0.000
R5 (Requiem for a Dream)	E11	0.000	0.000	1.000	0.000
S_{prop}		0.434	0.434	0.200	0.000
S_{class}		0.868		0.200	

Figure 5.15: Running Example: Voting for Schema Correspondences.

score is chosen is final class mapping \hat{C} . If multiple classes have the same score, the most specific class according to the type hierarchy in the knowledge base is chosen. Based on the final class decision, all candidate entities which are not an instance of the chosen class or one of its sub-classes are removed.

$$S_{class}(c) = \sum_{p \in P} S_{prop}(c, p) \cdot P_{class}(p, c) \quad (5.9)$$

$$\hat{C} = \arg \max_{c \in C} S_{class}(c) \quad (5.10)$$

In the running example, the voting scores are calculated as shown in Figure 5.15 under consideration of only the best entity candidate for each row. This results in an aggregated score of 0.434 for the `dbo:director` property and 0.0 for the `dbo:artist` property. The final class score is then obtained by summing up the scores for all properties, resulting in a score of 0.868 for `Film` and 0.2 for `Album`, i.e., `Film` is chosen as the final class.

4. Iterative Matching

The final step of the algorithm iterates between re-scoring of the entity and property correspondences. Entity candidates are re-scored by weighting the instance-based similarities with the property similarities. These scores are then aggregated into the average over all candidate properties \hat{P} and form the new entity candidate similarity score. Property correspondences with high similarity score hence have a larger influence on the candidate's similarity score. For the subject column and the `rdfs:label` property, the property similarity is replaced by a constant key-weight factor, which is a parameter of the algorithm and can be used to control the influence of the subject column on the candidate similarity. By increasing this factor, the subject column values gain higher importance for the candidate similarity.

Row	Entity	Title Score	Director Score	S_{cand}
R1 (Groundhog Day)	E1	0.670	0.434	0.552
R2 (Duck Soup)	E3	0.500	0.434	0.467
	E4	0.500	0.000	0.250
R4 (Crouching Tiger, Hidden Dragon)	E7	1.000	0.434	0.717
R5 (Requiem for a Dream)	E10	0.800	0.217	0.509

Figure 5.16: Running Example: Iterative Update.

$$S_{cand}(r, e) = \frac{1}{|\hat{P}|} \sum_{p \in \hat{P}} S_{prop}(\hat{C}, p) \cdot S_{inst}(r, e, \hat{C}, p) \quad (5.11)$$

After the re-scoring of the entity candidates, the property similarity scores are re-calculated as described in step 3. These new similarity scores \hat{S}_{prop} are then combined with the scores from the previous iteration as shown in equation 5.12, where λ is a parameter of the algorithm that controls the magnitude of the updates in each iteration.

$$S_{prop} = \lambda \hat{S}_{prop} + (1 - \lambda) S_{prop} \quad (5.12)$$

After a configurable number of iterations, the similarity scores are pruned using a top-1 global matching, i.e., for each column and each row, the property and entity candidate with the highest score is kept, all other scores are set to 0.

For the running example, the entity candidate scores are re-calculated as shown in Figure 5.16. The instance-based similarity scores from step 2 are multiplied with the property scores calculated in step 3 (shown in the “*Director Score*” column, and average over all properties (here for the “*Title*” and “*Director*” columns), resulting in new candidate scores. The key-weight is set to 1.0, so the scores for the “*Title*” column remain unchanged.

The new candidate scores can in turn be used for updates to the property scores and additional iterations. After finishing the iterations, a top-1 global matching is applied. For the relation annotation task, this results in a correspondences between the “*Title*” column and `rdfs:label` property as well as the “*Director*” column and the `dbo:director` property. For the entity annotation task, the correct annotations for the rows R1, R2, R4, and R5 to entities E1, E3, E7, and E10, respectively, are created, while R3 is not linked to any entity, because no corresponding entity exists in the knowledge base.

5.5 Experiments

This section presents the experimental evaluation of T2K Match on the T2D Gold Standard. First, the used subset of DBpedia and the parameter setting used for the experiments are described. Afterwards, the evaluation of T2K Match is presented and the results are compared with two baseline approaches and recently published related work.

Target Knowledge Base. The target knowledge base is a subset of DBpedia version 2014. It consists of all classes and properties from the ontology namespace³ which contain at least 1 000 entities and are located in the first four levels of the class hierarchy. Further, properties are excluded if they are used for less than 5% of the entities of their domain. This subset contains a total of 94 classes, 1 339 properties, and 3 million entities. Through the use of this subset the evaluation of the algorithm is focussed on its discriminative performance for the large, high-level classes in DBpedia, which is of greater importance for the large-scale profiling of web tables that will be discussed in Chapter 6 than its ability to make very fine-grained class decisions, for example by differentiating between an `AmericanFootballPlayer` and a `CanadianFootballPlayer`. The choice of DBpedia as target knowledge base is in part made for the same reason, as its type hierarchy is already much coarser than, for example, the almost half a million classes in the YAGO knowledge base. Otherwise, there are no indications in the literature that the choice of knowledge base has an impact on the performance of the algorithm. A comparative evaluation using DBpedia and YAGO was performed by Mulwad et al. [Mulwad et al., 2013], showing that their method performed slightly better with YAGO, but the authors did not further analyse the reasons. It is hence assumed that the choice of knowledge base can be made based on other criteria than its expected influence on the algorithm’s performance. As DBpedia is a central hub in the Web of Data (see Chapter 3), it is considered a good choice, as any web table that is linked to DBpedia then also has many connections into the Web of Data.

5.5.1 Parameter Optimisation

This section describes how the parameters of the T2K Match algorithm are determined. T2K Match defines 20 parameters, which include similarity functions for different data types, the top k parameters for candidates or weights for properties. The parameter setting used in the experiments is determined by a genetic algorithm and split validation on the T2D gold standard. Two equally sized subsets of the T2D instance-level gold standard are created by stratified sampling on the class mapping. The first subset is used to determine the parameter setting, while the second subset is used to evaluate the algorithm and measure its performance.

³<http://dbpedia.org/ontology/>

The resulting parameter set specifies Jaccard similarity for the candidate selection step with a k of 50, and a k of 100 for the candidate refinement step. The instance-based similarity functions are generalised Jaccard with Levenshtein similarity for token comparisons for the data type `string`, the ratio between absolute values for `numeric` values, and the difference in years for `date` values. The full parameter set is available on the T2K Match project website.⁴

5.5.2 Matcher Evaluation

This section presents the evaluation of T2K Match on the T2D gold standard, an error analysis, and a quantitative analysis for the different high-level classes in the DBpedia type hierarchy.

The results of T2K Match on the instance-level gold standard are shown in Table 5.1. The column “*F1 (train)*” indicates the F1-Score that was achieved on the subset that was used to tune the parameters, all other results indicate the micro-averaged performance on the evaluation subset. The three rows show the results for the entity annotation, relation annotation, and class annotation task, respectively.

Table 5.1: T2K Match Evaluation.

Task	Precision	Recall	F1	F1 (train)
Entity	0.90	0.76	0.82	0.86
Relation	0.77	0.65	0.70	0.73
Class	0.94	0.94	0.94	0.97

The evaluation shows a high precision 0.9 for the entity annotation task, with an F-measure of 0.82, an F-measure of 0.7 for the relation annotation task, and an F-measure of 0.94 for the class annotation task. There is only a small difference between the performance on the training set to the performance of the test set, which indicates that over-fitting is not an issue.

Errors for entity annotation are often made due to ambiguous names in web tables where no additional properties from the knowledge base exist. For example, for the class `Company`, the precision for tables where no properties beside `rdfs:label` can be mapped is only 0.45, but for tables where one additional property correspondence is found, it increases to 0.97.

For the relation annotation task, differences in the values stated in DBpedia and the web tables are an issue. Examples include varying weights of athletes or the birth dates of historic persons, which are often different from source to source. Another problem are ambiguities in the values of DBpedia properties. For example, a web table might state the language that is spoken in a country as “*German*”, which is more similar to the property `demonym` (with value “*German*”) than to the correct property `language` that has the value “*German language*”. These

⁴<http://web.informatik.uni-mannheim.de/T2K/parametersT2KMatch.csv>



Figure 5.17: F1-measure for different high-level classes.

issues indicate that specific similarity measures for each property in the knowledge base or additional sources of surface forms could improve the results.

Figure 5.17 shows a break-down of the F1-measure achieved for the different annotation tasks for the high-level classes in the DBpedia type hierarchy. The bars indicate the achieved F1-measure for the entity, relation, and class annotation task, respectively, and the line indicates the percentage of web tables that are assigned the classes indicated by the groups. The chart shows that some classes are harder to match than others, which can be for a variety of reasons such as the examples mentioned earlier. These differences are an indication that a specialised configuration of the matcher per class could improve the results. However, this would increase the requirements to the training set, meaning that larger amounts of web tables need to be annotated. Figure 5.18 further displays the achieved performance by table size and shows a tendency towards better performance for large tables and that the worst performance is achieved for very small tables with up to ten rows.

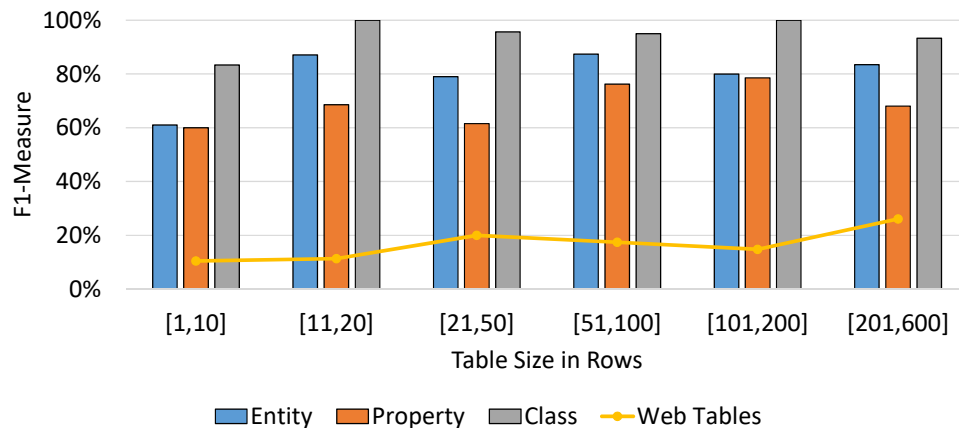


Figure 5.18: F1-measure for different table sizes.

Closer inspection of the errors made by the algorithm reveal a variety of different challenges that a semantic table interpretation method has to deal with. Some properties are hard to distinguish based on the instance values. For example, airports in the United States often have the same value for their IATA and FAA code, which makes these properties indistinguishable based on instance-based similarities for web tables that only contain such airports. Table 5.2 shows another example: a table with two population columns with very similar values, which make it hard to determine that the first of these columns refers to the total population of the city and the second refers to the population of the metropolitan area of the city. Another problem is the sparsity of the properties in DBpedia: As many properties have no value for most of the entities, often no instance-based similarity value can be calculated.

Table 5.2: Example of a web table that is hard to match due to very similar values in two columns with different semantics.

City	Country	Population (1)	Population (2)
Ecatepec	Mexico	1 688 000	1 690 000
Haiphong	Vietnam	1 885 000	1 885 000
Hubli	India	801 000	801 000
Incheon	South Korea	2 630 000	2 630 000
Indore	India	1 912 000	1 920 000
Luoyang	China	1 500 000	1 499 000

For the disambiguation of entities and the class annotation, the most challenging issue are web tables that do not contain any attributes that exist in the knowledge base except the subject column. In these cases, only the entity names in the subject column can be used by the algorithm. Table 5.3 shows such a web table which is hard to match as the entity names in the first column are highly ambiguous and no additional properties are contained that could be used.

Table 5.3: Example of a web table that is hard to match due to ambiguous entity names and no additional properties that exist in the knowledge base.

Name of Attraction	Category	Visitor 2009	Visitor 2010
British Museum	Museums	5569981	5842000
National Gallery	Museums	4780030	4954914
Natural History Museum	Museums	4105106	4647613
Science Museum	Museums	2753493	2757917
National Railway Museum	Museums	767863	619952

5.5.3 Comparison with other Approaches

This section compares the results achieved by T2K Match to two baseline methods, two open-source matching systems as well as two semantic table interpretation methods that were published after the original publication of T2K Match. The first baseline is the initial candidate selection step of T2K Match, and shows the improvements of the additional components of the algorithm. The second baseline is the lookup service provided by DBpedia. The two matching systems *LogMap* and *PARIS* are available as open-source and have been evaluated by Efthymiou et al. [Efthymiou et al., 2017] on the T2D gold standard. Further, the method proposed by Efthymiou et al. and T2K Match++, an extension to T2K Match proposed by Ritze [Ritze, 2017], are compared. Each of these methods is briefly described in the following.

Candidate Selection. The candidate with the highest score for each row after the first index lookup of the candidate selection is chosen as annotation. This baseline serves as a reference value to gauge the improvements achieved by the following steps of the T2K Match algorithm.

DBpedia Lookup. The DBpedia lookup service⁵ is queried for each value in the subject column and the first result is chosen as annotation. In addition to relying on string comparison, this service considers the hyperlink in-degrees of the corresponding Wikipedia page of each entity and prefers more popular matches.

LogMap. The LogMap matcher is designed for ontology matching [Jiménez-Ruiz and Grau, 2011]. It relies on lexical and structural similarities of class hierarchies and employs an iterative discover-and-repair method to produce mappings that are free from logical inconsistencies. It is, however, not specifically designed for web tables and their challenges.

PARIS. The PARIS matcher uses probabilistic modelling to align the schema and entities of two ontologies simultaneously [Suchanek et al., 2011]. While it has been designed based on a carefully designed probabilistic model, its focus is on matching comparably clean ontologies rather than noisy web tables. For this method, only exact value comparisons can be used, as it does not scale to the size of the DBpedia knowledge base for other similarity measures.

[Efthymiou et al., 2017]. Efthymiou et al. propose a hybrid method that uses index lookups for entity labels from Wikidata as well as entity embeddings, which are vector representations of entities which are obtained by using random walks in the knowledge base and applying the word2vec model [Mikolov et al., 2013].

⁵<http://wiki.dbpedia.org/lookup/>

T2K Match++. Ritze extends T2K Match with additional features, a supervised classification model, and a holistic matching step. These additions result in an increased performance over the original algorithm and represent the current state of the art in semantic table interpretation [Ritze, 2017].

The comparison is summarised in Table 5.4 and shows that the components of T2K Match result in a large improvement in precision and recall over the initial candidate selection, with an improvement of 0.37 in precision and 0.23 in recall, as well as over the DBpedia lookup service, with an improvement of 0.11 in precision and 0.03 in recall. The DBpedia lookup service and T2K Match use similar mechanism to improve recall, so their difference with regard to this measure is small, but the exploitation of additional columns beside the entity label results in a strong improvement in precision for T2K Match.

The two open source-matchers LogMap and PARIS cannot beat the DBpedia lookup service baseline, as both methods result in a very low recall. This shows that web tables require specialised methods and cannot be understood semantically with existing ontology matching algorithms. Finally, the two methods that were proposed after the initial publication of T2K Match outperform its results by 0.03 and 0.05 points in F1-measure, respectively. The improvements of the method proposed by Efthymiou et al. increase recall through a combination of an index-based lookup and embedding-based methods. Their approach first performs an index lookup, and in case no match is found, the embedding vector closest to the entity mention from the web table is used. The improvements of T2K Match++ stem from the large number of additional features, and, according to the result analysis by Ritze, from the holistic matching which increases the confidence of the matcher for web tables where no properties can be mapped to the knowledge base.

Table 5.4: Comparison of different methods for the entity annotation task.

Method	Precision	Recall	F1-measure
Candidate Selection	0.53	0.53	0.53
DBpedia Lookup	0.79	0.73	0.76
LogMap	0.89	0.57	0.70
PARIS	0.42	0.04	0.07
T2K Match	0.90	0.76	0.82
[Efthymiou et al., 2017]	0.87	0.83	0.85
T2K Match++	0.94	0.80	0.87

5.6 Conclusion

This chapter presented the topic of semantic table interpretation and introduced T2K Match, an algorithm that solves the three semantic table interpretation tasks of annotating web tables with classes, columns with properties and rows with entities from a target knowledge base. The topic of semantic table interpretation and the scope of its tasks were defined and common approaches for each task were introduced and reviewed in Section 5.2. This revealed that both comparability and reproducibility are limited in this field, and the following sections introduced a new gold standard and a new algorithm, which both are made publicly available. The gold standard and the methodology and tools used to create it were presented in Section 5.3 along with detailed statistics about its contents. The proposed algorithm, T2K Match, was presented in Section 5.4 and experimentally evaluated in Section 5.5. Other than most of the algorithms proposed in the literature, it can annotate literal columns, which are ubiquitous in web tables, in addition to columns containing entity names. In the experimental evaluation, T2K Match was shown to improve the state of the art in semantic table interpretation and to produce comparable results to algorithms that were proposed in the following years.

This chapter made the following contributions:

- **Literature Review:** A comprehensive literature review that organises the common methods and approaches according to the different semantic table interpretation tasks as well as a summary of all relevant, published work about the topic. Further, the important issues of comparability and real-world applicability were discussed, showing the need for openly available datasets and algorithms.
- **T2D Gold Standard:** A large, publicly available gold standard containing web tables and annotations for the tasks of annotating web tables with classes, columns with properties, and rows with entities using DBpedia as target knowledge base. The schema-level gold standard contains 1748 web tables, of which 762 are content tables that are annotated with classes and a total of 2084 property annotations. Of these content tables, 233 are additionally annotated with 26 124 entity annotations. In addition to earlier gold standards, T2D also contains negative examples and annotations for literal columns, which allows for a more comprehensive evaluation that gives indication to the practical applicability of an algorithm.
- **T2K Match Algorithm** A novel algorithm for semantic table interpretation that solves the tasks of class, relation, and entity annotation. T2K Match achieves F1-measure values of 0.94 for class annotation, 0.82 for entity annotation, and 0.7 for relation annotation. The contributions are the annotation of literal columns as well as the improved quality. Even methods published after T2K Match only achieve small improvements of up to 0.05 in F1-measure for the entity annotation task, showing that it represents a strong benchmark on the T2D gold standard.

The literature review presented in this chapter has shown that, although many different approaches for semantic table interpretation have been proposed, comparability and real-world applicability are often limited. The different combinations of tasks that are solved and the different datasets that are used for evaluation hinder a comparison of the methods based on the results reported in the literature. The close-source nature of many methods further prevents an experimental comparison of the various approaches.

This situation has been improved by publishing the T2D gold standard and the open-source T2K Match algorithm. The gold standard contains annotations for all semantic table interpretation tasks that are required to generate triples for knowledge base augmentation from web tables and allows for a comparative evaluation of methods in this area. The algorithm further sets a benchmark result on this gold standard that other systems can be compared to and also serves as a basis for advanced methods that extend its open-source implementation [Ritze and Bizer, 2017]. With respect to practical applicability, the algorithm not only focuses on named entity columns, but also matches columns with other data types such as numbers or dates, which make up between 35 and 52% (see Chapter 4) of the columns in the used web table corpora and should hence not be ignored.

With the contributions in this chapter, future research in the area of semantic table interpretation has been enabled to comparatively analyse algorithms by reusing the published gold standard and algorithm [Efthymiou et al., 2017, Ritze and Bizer, 2017, Ermilov and Ngomo, 2016, Pham et al., 2016]. The comparison of the performance with approaches proposed after the original publication of T2K Match shows that only small gains in performance have been achieved by the more recent approaches, which indicates that T2K Match sets a strong benchmark on the T2D gold standard for other researchers to compare their methods to. Further, practical applicability is enhanced as the proposed algorithm can deal with more types of data than earlier approaches. This in turn enables the large-scale analysis of the contents of web table corpora, which will be presented in the next chapter.

Chapter 6

Corpus Profiling

6.1 Introduction

The availability of large-scale web table corpora promises a wealth of potentially useful information for many tasks, such as data search [Cafarella et al., 2009, Yakout et al., 2012], knowledge base augmentation [Limaye et al., 2010, Dong et al., 2014a], and question answering [Yin et al., 2011, Sarawagi and Chakrabarti, 2014]. However, existing studies focus on specific applications and do not profile the contents of the used web table corpora exhaustively. Most studies either use very small and thus not representative datasets or, if they are applied to large web table corpora, those are privately owned and little information about their content is published. This makes it impossible to generalise or reproduce the research results and prevents a more general judgement of the quality and topical coverage of the corpora as a whole. Existing work on the topical contents of web table corpora is limited to class-level information for large web tables [Hassanzadeh et al., 2015]. To fill this gap, this chapter presents a topical profile of the Web Data Commons Web Tables Corpus 2012 with respect to the DBpedia knowledge base. Using T2K Match as semantic table interpretation algorithm, every web table in the corpus is matched to the knowledge base and the distributions of correspondences to classes, entities, and properties are presented and discussed. Following this analysis, the quality of the extracted triples is experimentally validated using state-of-the-art data fusion methods.

The focus of this chapter is to analyse the potential of web tables for the slot-filling knowledge base augmentation task. Nowadays, cross-domain knowledge bases such as DBpedia, YAGO, or the Google Knowledge Graph are already employed as background knowledge in a wide range of different applications, including web search, question answering, data integration, and entity linking. For these applications, the content of the knowledge bases must be as complete, correct, and up-to-date as possible. Cross-domain knowledge bases must hence continuously be updated and extended using high-quality data from external sources, and this chapter analyses to which extent web table corpora can be used for this task.

Specifically, the coverage of different topics and their granularity is presented by reporting the distributions of correspondences between web tables and a target knowledge base over different classes, entities, and properties. It is shown that with the applied method, 949 thousand web tables can be mapped to known classes, with a total of 13.7 million correspondences to known entities and 562 thousand correspondences to known properties. For the slot-filling task, further experiments are reported that investigate the frequency of statements in the corpus, i.e., how many sources contain the same statement about the property value of an entity, as well as their quality. This shows that, although the majority of the extractable triples is incorrectly interpreted by current methods, a subset of high-quality triples can be created by knowledge-based trust data fusion methods.

The contributions of this chapter are the following:

- **Topical Profile:** An in-depth analysis of the correspondences created between web tables and the DBpedia knowledge base provides the most fine-grained topical profile of any web table corpus at the time of writing. The profile exceeds the scope of earlier studies and provides detailed insights into the contents of the corpus. The frequency distributions reveal that most recognised entities and relations are only found in very few web tables, while a small set of head entities and properties is discovered very frequently.
- **Data Type Profile:** By analysing the data type distribution for the matched columns, it is demonstrated that the majority of extractable statements from web tables contain numerical values. This reveals a shortcoming of many approaches for semantic table interpretation which focus exclusively on the annotation of named entity columns.
- **Quality of Extracted Triples:** Using the local-closed-world assumption for distant supervision, the quality of all extracted triples is evaluated. The results show that although the overall quality of the raw triples after extraction is low, a set of high quality triples can be determined. Profiling these high-quality triples shows which classes and properties can be augmented with high quality from the web tables corpus.

This chapter is organised as follows. First, related work in the areas of web data profiling and web data fusion is introduced in Section 6.2. Section 6.3 then presents the topical profile for web tables in the Web Data Commons Web Tables Corpus 2012. Afterwards, the quality of the extractable triples is analysed in Section 6.4.

The work presented in this chapter, i.e., the large-scale data profile of the WTC 2012, has previously been published in [Ritze et al., 2016].

6.2 Related Work

Several studies have shown that knowledge extracted from web tables can be useful for applications like table search [Venetis et al., 2011, Balakrishnan et al., 2015], table extension [Yakout et al., 2012, Lehmborg et al., 2015, Das Sarma et al., 2012], and knowledge base augmentation [Wang et al., 2012, Dong et al., 2014a, Sekhavat et al., 2014]. For most of these applications, the performance of matching methods for web tables plays an important role [Zhang et al., 2013, Limaye et al., 2010, Yakout et al., 2012, Braunschweig et al., 2015] and is the focus of the respective publications. However, in order to judge the potential of web tables for different applications, it is essential to have an understanding of the data profile and topical distribution of large Web table corpora [Hassanzadeh et al., 2015]. This section introduces related work about methods for profiling and their results for web tables as well as related research areas.

Web Page Profiling. The structural properties of the web graph have been profiled by Broder et al. [Broder et al., 2000], who analyse the distribution of in- and out-degree of web pages, i.e., the number of ingoing and outgoing hyperlinks. They find that both distributions follow a power-law distribution, in which few web pages have a very large number of in- or outgoing hyperlinks, while the majority of web sites has only few links. Later, Broder’s analysis was updated based on a larger crawl of the web and some of their findings about the specific degree distributions were revised [Meusel et al., 2014], but the general trend of few very large web pages could still be confirmed. Similar distributions can be found for the frequency of entities in the web tables, where some very popular entities occur very frequently, while the majority of entities only occurs a few times.

Linked Data Profiling. For the Linked Data Web, Hogan et al. [Hogan et al., 2012] as well as Schmachtenberg et al. [Schmachtenberg et al., 2014] focus on the adoption of the linked data best practices (linking to other datasets, re-using vocabularies, and providing metadata), but they also provide a topical analysis. Hogan et al. report statistics about a crawl in 2010 of ca. 1 billion statements obtained from almost 800 data providers. They find that most statements assert type membership or facts about people, with the most frequent predicate being `rdfs:type` (18.5%) and the most frequent type being `foaf:Person` (79.2%). This high frequency of data about people can also be observed for the web tables in the WTC 2012, where the most frequently found class is `dbo:Person`.

Web Table Profiling. Hassanzadeh et al. [Hassanzadeh et al., 2015] analyse the topical distribution of the 90 million tables in the WDC Web Tables Corpus 2012 by matching columns to classes of different knowledge bases. By comparing the web tables to DBpedia, YAGO and Schema.org data, they show that the size and topics that are covered by the knowledge base influence the distribution of correspon-

dences that are discovered through matching. For DBpedia, they find that the most common classes are `Agent`, `Person`, and `Place`. For YAGO, the most frequent classes are `PhysicalEntity`, `Object`, and `YagoLegalActorGeo`. For `schema.org`, the most common classes are `Person`, `Place`, `CreativeWork`. They further find that only a relatively small fraction of the web tables can be matched to the knowledge bases. They are able to map 1.7 million columns to DBpedia, 500 thousand columns to `schema.org`, 6.3 million columns to YAGO, and 26.6 million columns to Freebase. For this analysis, they fix the minimum overlap score between the values in a column and the labels in the knowledge base to 20, which indicates that only large web tables, which have at least 20 rows, are considered. This means that the majority of the web tables in the corpus, which have less than 20 rows, are not analysed. Beside the work of Hassanzadeh et al., no profiling for web tables on the semantic level has been published. However, there are structural profiles of web tables, for example for different table types, as discussed in Chapter 4.

Web Table Data Fusion. In the context of this chapter, data fusion is used to estimate the quality of the data that can be extracted from web tables and represents an additional dimension for data profiling. For the large-scale evaluation of statements that are extracted from web sources, Dong et al. [Dong et al., 2014b] propose the Local-Closed-World Assumption (LCWA). Under this assumption, a knowledge base is considered *locally complete* and used to determine the accuracy of facts. Specifically, all triples contained in the knowledge base are considered true and all triples which are not contained in the knowledge base are considered false, if the knowledge base contains at least one triple for the same subject and predicate combination. Using the LCWA they evaluate the quality of extracted triples from 12 different extractors and over 1 billion web pages before data fusion in two different studies and find an overall accuracy of only 30% [Dong et al., 2014b] and 11.5% [Dong et al., 2014a]. A detailed error analysis, however, reveals that only 4% of the errors are due to source incorrectness, the remaining errors are introduced by the extractors. Their approach to use the data overlap with a knowledge base to estimate source reliability, termed knowledge-based trust, as well as the evaluation using the LCWA are the foundations of the data fusion methodology applied in this chapter.

In summary, the profiling of the contents of web tables has not received much attention in the research community yet, although many approaches for semantic table interpretation and data fusion of data from web tables haven been proposed. Further, the profiling of related web data sources such as linked data or the hyper-link graph of the Web shows certain similarities concerning the topical or structural distributions.

6.3 A Topical Profile for Web Tables

In this section, a topical profile for web tables based on the Web Data Commons Web Tables Corpus 2012 is presented. This profile is created by matching all web tables in the corpus to the DBpedia knowledge base using the T2K Match algorithm and shows the distribution of topics represented by the classes, entities and properties in DBpedia.

6.3.1 Methods

This section describes the methodology that is used to create the topical profile. First, T2K Match, and its extension for the creation of the profile are described. Then, the knowledge base to which the web tables are matched and the used corpus of web tables are introduced.

Semantic Table Interpretation Algorithm. T2K Match is an algorithm that maps web tables to a knowledge base by solving three matching tasks: (1) it maps a web table to a class in the knowledge base that contains the entities which are described in the web table; (2) it maps the columns of a web table to properties in the knowledge base that have the same semantic meaning; (3) it maps the rows of a web table to entities in the knowledge base that represent the same real-world entity. The T2K Match algorithm is described in detail Chapter 5. In brief, the method initially determines a set of candidate entities for the rows in the web table. Based on these candidates, the algorithm decides for the corresponding class and calculates instance-based similarity scores using data-type-specific similarity metrics. Using these scores, the algorithm iteratively refines the property annotations for columns and entity annotations for rows. During the whole process, the data and schema matching steps mutually influence each other, as the result of each matching step is used to calculate the scores of the next matching step.

In addition to the T2K Match method described in Chapter 5, the method applied here not only annotates entities for the subject column of a web table, but also for all other columns which are mapped to an object property, i.e., a property which has entities instead of literals as its range. The original T2K Match algorithm annotates columns with such properties by comparing their values to the `rdfs:label` of the corresponding entities in the knowledge base, but does not annotate the cells with these entities. All statistics in this chapter will refer to such columns as data type `reference`.

Target Knowledge Base. The target knowledge base for the topical profile is DBpedia 2014.¹ It describes 4 584 616 instances using 2 795 different properties and 685 classes. This set of classes, properties, and instances limits the scope of the created profile, as T2K Match cannot map web tables to topics which are not

¹<http://wiki.dbpedia.org/data-set-2014>

Table 6.1: Characteristics of the analysed web table corpus.

	Numeric	Date	String	μ	Σ
Columns	46M	4M	86M	4.122	137M
Rows	-	-	-	21.499	716.6M
Values	995M	101M	1.9B	88.611	2.95B

covered by the knowledge base. As indicated by Hassanzadeh et al. [Hassanzadeh et al., 2015], the coverage of entities for the classes in the knowledge base affects the profiling results as frequent classes in the knowledge base are likely also frequent in the profile. This frequency distribution of entities in the knowledge base is shown as the series “*DBpedia Entities*” in Figure 6.1 for the four largest top-level classes and their two largest sub-classes, respectively.

Web Table Corpus. The analysed web table corpus is the English-language subset of the Web Data Commons Web Tables Corpus 2012 (see Section 4.4.1). The English-language subset contains 91 million web tables from the five top-level domains .com, .org, .net, .eu, and .uk. After recovering the table metadata (see Section 4.3), 33.3 million web tables remain for which a subject column could be detected. Table 6.1 shows statistics about the number of columns, rows, values, and detected data types in the web tables. The web tables in this subset are larger than the web tables in the full corpus, with an average of 21.5 rows and 4.1 columns compared to 12.4 rows and 3.5 columns in the complete corpus. The distribution of the detected data types is comparable to the distribution in the full corpus, with an increase in numeric columns from 25% to 34%. The tables in this corpus originate from 97 932 different websites and Table 6.2 shows the most frequent web sites and column headers to provide an initial overview over the covered contents. Frequent topics among these web sites include music (apple.com, 7digital.com), sports (baseball-reference.com, latestflnews.com), and shopping (amazon.com, inkjetsuperstore.com). The column headers also indicate a high frequency of the topics shopping (“5 star”, “price”), music (“album”, “artist”), and sports (“team”, “pos”).

6.3.2 Correspondence Statistics

This section reports the results of applying T2K Match to the Web Data Commons Web Tables Corpus 2012. Specifically, the set of matched tables, correspondences to classes, entities, and properties, as well as the distribution of data types for matched properties and the frequency with which different sources provide the same value for different entity and property combinations are discussed.

Table 6.2: Analysed Corpus: Most frequent web sites and column headers.

Web Site	Tables	Header	Tables
apple.com	50 910	<empty>	14 495 456
patrickoborn.com	45 500	5 star:	2 402 376
baseball-reference.com	25 647	name	1 813 064
latestflnews.com	17 726	price	1 771 361
nascar.com	17 465	date	1 603 938
amazon.com	16 551	amazon price	1 178 559
baseballprospectus.com	16 244	formats	1 066 836
wikipedia.org	13 993	title	913 260
inkjetsuperstore.com	12 282	1	897 107
flightmemory.com	8 044	time	856 401
sportfanatic.net	7 596	description	773 883
tennisguru.net	7 504	size	692 251
windshieldguy.com	7 305	replies	605 075
donberg-electronique.com	6 734	used from	589 278
citytowninfo.com	6 293	new from	589 259
juggle.com	5 752	year	579 726
deadline.com	5 274	location	546 856
blogspot.com	4 762	album	526 375
7digital.com	4 462	type	501 747
electronic-spare-parts.com	4 421	#	500 198
angielskapilka.com	4 379	s	465 256
allfreemp3.net	4 250	-1	454 121
goal.com	4 008	:	451 557
hookedgamers.com	3 944	latest post	421 737
fullgasrevista.com	3 783	discussion	412 672
drugs-about.com	3 727	artist	390 805
motortrend.com	3 711	model	385 354
wn.com	3 621	rank	378 570
reddit.com	3 613	0	346 029
faz.net	3 424	<encoding error>	341 751
racingwest.com	3 383	trade name	328 856
go.com	3 236	active ingredients	328 228
5pmusic.com	3 155	pharmaceutical company	328 179
leadlinkmedia.com	3 126	city	315 139
wordpress.com	3 121	<encoding error>	306 884
lechita.net	3 118	rating	295 700
zipglassnetwork.com	3 095	team	292 352
prostarautoglassllc.com	3 083	m	282 616
carsdirect.com	3 034	pos	267 543
meilleur-artisan.com	2 969	age	257 594

A summary of all results is given in Table 6.3, which shows aggregated statistics for the high-level classes in the DBpedia type hierarchy. Column “ T_e ” states the size of the set of tables for which at least the subject column and thus a set of entities could be matched to DBpedia. Column “ T_p ” covers all tables which in addition have a property correspondence. Column “ V ” shows the amount of cells (values) contained in tables of T_p for which an entity annotation for the corresponding row and a property annotation for the corresponding column exists. In other words, V expresses how many triples can be generated from the tables. These numbers are further divided by data type in the last four columns of the table.

Tables. The result of matching the 33.3 million web tables in the corpus to the DBpedia knowledge base is a mapping for 949 970 web tables (2.85%). This means that only a small fraction of the web tables in the corpus can be understood with the classes, entities, and properties that exist in the DBpedia knowledge base and limits the coverage of the data profile that is presented in this chapter. This small fraction of mapped tables indicates that (1) the topical overlap between the web tables and the knowledge base is low and/or (2) many of the relational tables are not entity-attribute tables and can hence not be mapped. Low topical overlap is also indicated by the frequency of product-related column headers like “*5-star*” (2.4 million tables) or “*price*” (1.7 million tables) in the corpus (see Table 6.2), which cannot be mapped as DBpedia does not cover specific products. A similarly low overlap was also reported by Venetis et al. [Venetis et al., 2011], who were able to annotate 185 thousand web tables (1.5%) with the YAGO knowledge base and 577 thousand web tables (4.7%) with the Freebase knowledge base using a corpus of 12.3 million web tables.

The web tables that qualify for the mapping must have a subject column in which at least 25% of the rows can be mapped to known entities from DBpedia. The 949 thousand web tables that satisfy this condition are potentially useful for the set-completion and schema-extensions tasks. Set completion can make use of the rows which have not been mapped to the knowledge base, as they potentially contain unknown entities. Schema extension can exploit the unmapped columns to create new properties. The web tables that additionally have at least one column mapped to a property in the knowledge base are further useful for the slot-filling task. This stricter condition is satisfied by 301 450 tables (Column “ T_p ” in Table 6.3) and contain a total of 8 million values (Column V in Table 6.3) for which a knowledge base triple can be generated.

Table 6.3: Aggregated correspondence statistics. Each row represents the aggregated statistics for the class mentioned in the first column and all its subclasses.

Class	Tables		Values		Data Types				Groups	
	T_e	T_p	V	V/T_p	Numeric	Date	String	Reference	G	V/G
Agent	460 150	140 221	4 751 275	33.88	2 217 617	1 776 739	367 338	389 581	454 016	10.46
Person	265 685	103 801	4 176 370	40.23	2 117 793	1 588 475	266 628	203 474	366 048	11.41
Athlete	243 322	95 916	3 861 641	40.26	2 084 017	1 435 775	163 771	178 078	284 213	13.59
Artist	9 981	2 356	18 886	8.02	3	11 527	3 499	3 857	6 842	2.76
Politician	3 701	1 388	18 505	13.33	10	7 725	3 393	7 377	6 559	2.82
OfficeHolder	2 178	1 435	131 633	91.73	30	66 762	59 332	5 509	11 362	11.59
Organisation	194 317	36 402	573 633	15.76	99 714	187 370	100 710	185 839	87 527	6.55
Company	97 891	6 943	203 899	29.37	58 621	83 001	34 665	27 612	25 164	8.10
SportsTeam	50 043	2 722	31 866	11.71	2 206	22 368	43	7 249	2 453	12.99
Edu.Inst.	25 737	14 415	238 365	16.54	38 056	64 578	13 334	122 397	35 736	6.67
Broadcaster	14 515	11 315	93 042	8.22	564	13 095	52 186	27 197	21 687	4.29
Work	269 570	127 677	2 284 916	17.90	109 265	1 354 923	33 091	787 637	331 071	6.90
MusicalWork	138 676	80 880	1 131 167	13.99	64 545	396 940	7 610	662 072	201 186	5.62
Film	43 163	9 725	256 425	26.37	10 844	198 913	14 382	32 286	56 610	4.53
Software	39 382	23 829	486 868	20.43	418	414 092	9 194	63 164	33 552	14.51
Place	133 141	24 341	859 995	35.33	413 375	273 510	84 111	88 999	100 673	8.54
PopulatedPlace	119 361	21 486	787 854	36.67	405 406	257 780	57 064	67 604	71 981	10.95
Country	36 009	6 556	208 886	31.86	93 107	66 492	31 793	17 494	5 709	36.59
Settlement	17 388	2 672	17 585	6.58	4 492	6 662	2 444	3 987	1 879	9.36
Region	12 109	427	5 625	13.17	3 097	897	292	1 339	1 193	4.72
Architect.Struct.	10 136	1 815	46 067	25.38	3 976	7 387	23 110	11 594	17 697	2.60
NaturalPlace	1 704	254	2 568	10.11	866	696	340	666	1 203	2.13
Species	14 247	4 893	83 359	17.04	-	7 902	38 682	36 775	23 809	3.50
Total	949 970	301 450	8 037 562	26.66	2 751 105	3 437 420	536 526	1 312 511	929 170	8.65

Classes. Figure 6.1 shows the distribution of class and entity correspondences for the four largest top-level classes in DBpedia and their two largest sub-classes, respectively. Almost 50% of all matched web tables have been mapped to the `Agent` class, which describes people and organisations and is also the largest class in DBpedia. The second-most frequently matched class is `Work`, followed by `Place`. Comparing the relative frequency of entities in DBpedia and web tables for the top-level classes, it can be seen that `Agent` and `Place` are similarly distributed, but `Species` and `Work` are not. There are only few web tables which are mapped to the `Species` class, which indicates that this topic is under-represented in the analysed corpus. The high frequency of web tables that are mapped to the `Work` class indicates that this is an important topic in the corpus.

Entities. In total, 13 726 582 entity correspondences for 717 174 unique entities are created, which is 15.6% of all entities in DBpedia. Figure 6.1 shows that the number of web tables mapped to a class and the number of rows mapped to entities of this class are similarly distributed. An exception is the class `Species`, which exhibits more entity correspondences than class correspondences. This indicates fewer mapped tables with more mapped entities per table. Figure 6.3 shows that for 70% of all entities, at least two web tables are found and 25% of the entities are contained in ten or more web tables. This is an important information for the slot-filling task: while there are many sources for some entities, which makes it easier to find missing values, the majority of the entities is only described in very few sources, which complicates finding a correct values which are missing in the knowledge base.

Properties. Figure 6.1 shows the distribution of class and property correspondences. More than half of all property correspondences were discovered for the `Person` class (52%), followed by the `MusicalWork` class (19%). Aggregated over all tables, a total of 562 445 property correspondences for 721 unique properties is found. Other than entity correspondences, property correspondences have a different distribution than class correspondences. For the `Agent` class, more property correspondences than class correspondences are found, which indicates that in many of these web tables, multiple columns represent a property that exists in the knowledge base. For the `Place` class, fewer property correspondences are found, which indicates that in many of these web tables, only few or even no columns represent a property that exists in the knowledge base. Figure 6.3 shows that 88% of all discovered properties occur in at least two web tables and 60% occur in at least ten web tables. Compared to the distribution of entity correspondences, properties occur much more frequently in different web tables than entities.

Examples. Table 6.5 lists the most frequent classes, properties, and entities in the created correspondences. Most of these entities belong to subclasses of the `Agent` class: `Company` and `Person`. The table further reveals that the five

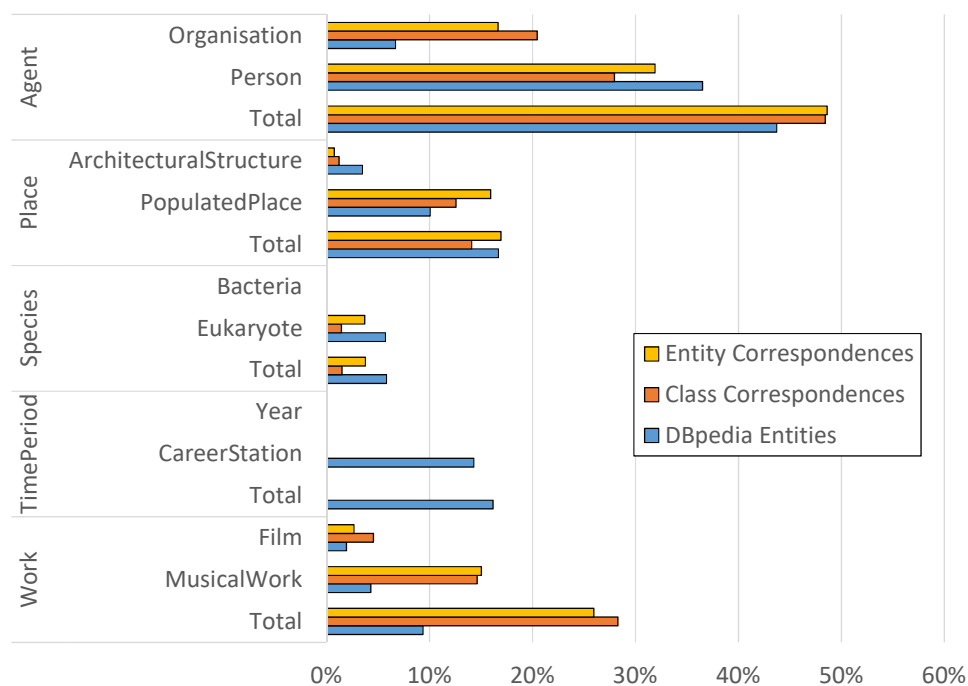


Figure 6.1: Distribution of DBpedia entities and correspondences for matched web tables by class.

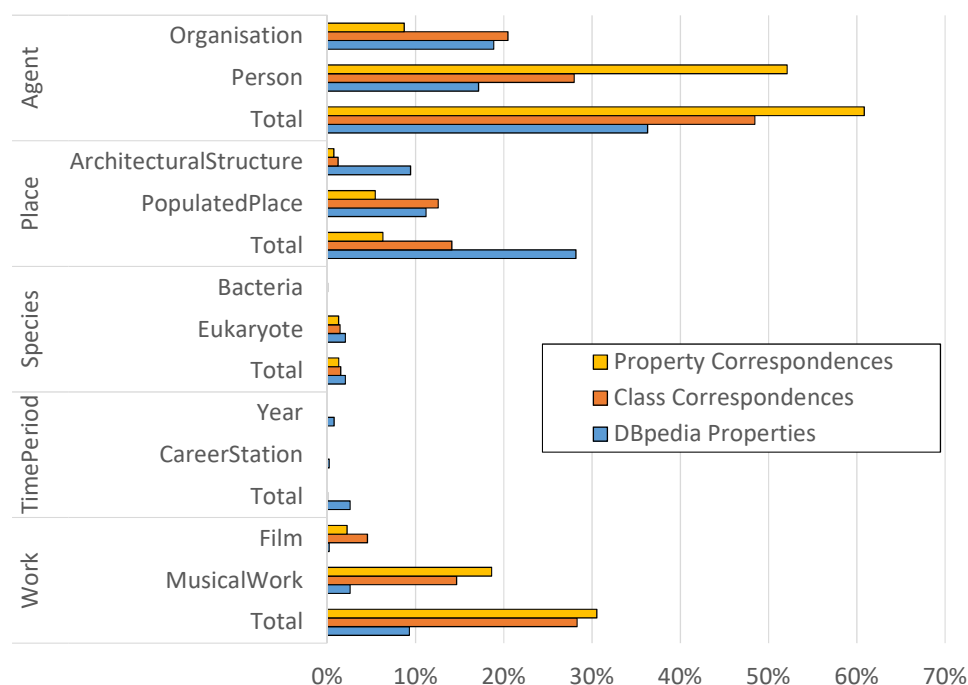


Figure 6.2: Distribution of DBpedia properties and correspondences for matched web tables by class.

most-frequent entities occur in about half of all web tables that are mapped to the *Company* class. A possible explanation is that these tables show different statistics, possibly at different points in time, for the same set of entities rather than just listing basic information for many different entities. The fact that no property was mapped equally often indicates that the attributes provided by these web tables do not exist in the knowledge base. The most frequently matched property, *statisticValue*, is due to many baseball statistic websites in the corpus, which provide web tables with various statistics about players' performances during different seasons and matches. As this property is very broadly defined in the knowledge base, its usefulness for the slot-filling task very limited, as different statistics are mapped to this property.

Group Size Distribution. For each combination of an entity correspondence and a property correspondence in the same web table, a triple, consisting of the entity, the property, and the respective value, can be extracted. By grouping these triples by their entity and property combination, a set of all discovered values for this combination can be obtained. Grouping the 8 million triples that can be generated from the correspondences results in 929 170 groups. Figure 6.4 shows the distribution of groups by size. Of all groups, 58% contain triples from at least two sources and 39% from at least three sources. Triples from ten or more sources can be found for 13% of all groups. Very frequent groups, which are supported by at least 100 sources, constitute 1% of all groups. This shows that although 25% of all instances and 60% of all properties were found in more than ten web tables, their combinations are much less frequent. For 42% of all groups, only a single triple can be extracted. Column “*G*” in Table 6.3 shows the number of groups by class and column “*V/G*” shows the average group size.

Data Types. The distribution of the 8 million values over the different data types shows that the majority of the extractable values is of type *date* (43%), followed by *numeric* (34%) and *reference* (16%). Half of the date values and more than two thirds of the numeric values are found for the class *Athlete*. Compared to the distribution of data types on the complete corpus, there is a shift from *string* values (64% before) to *date* (3% before) and *numerical* values (33% before). This indicates, for the columns that could be matched to the knowledge base, that web tables tend to contain more factual data in the form of dates and numbers than textual descriptions or references to other entities. Table 6.5 supports this by showing that the most frequent properties are numerical and date-valued properties. Table 6.4 compares the distribution of values by data type for the web tables corpus, the triples generated from the correspondences, and the groups generated from these triples. The largest groups are numeric values, with an average of 13.59 values per group, and the smallest for references, with 5.02 values per group.

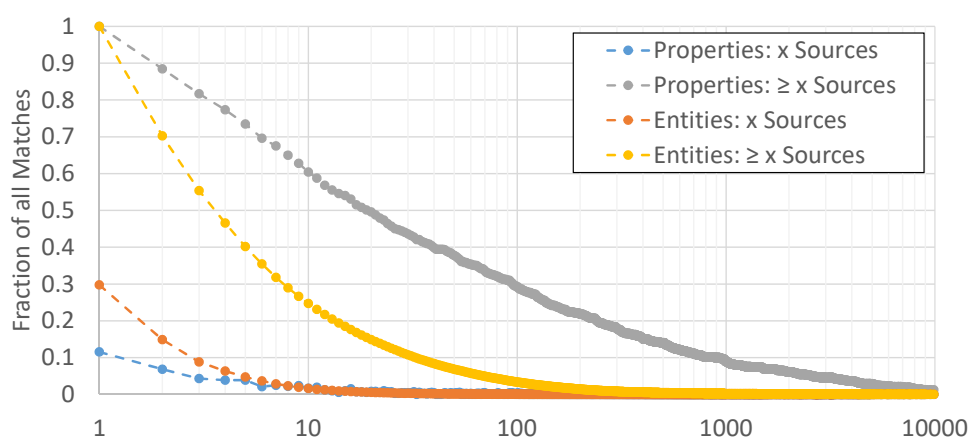


Figure 6.3: Distribution of correspondences by number of web tables.

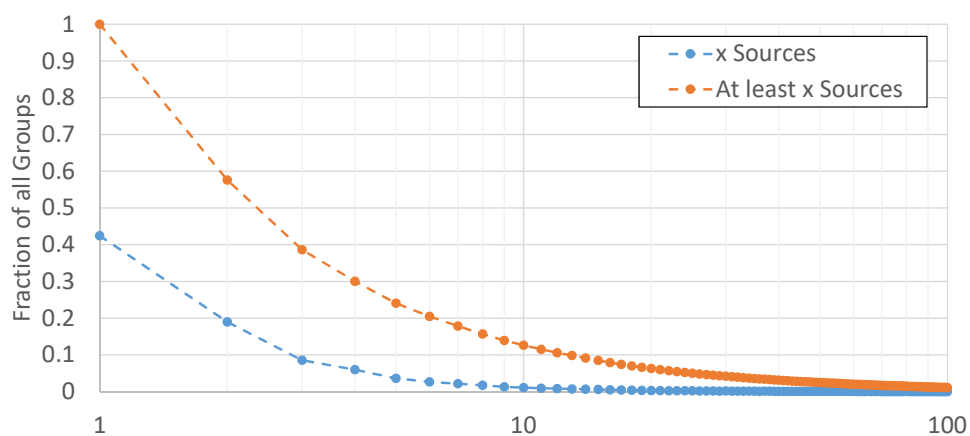


Figure 6.4: Distribution of group sizes.

Table 6.4: Distribution of values by data type.

Data Type	Corpus	Triples V	Groups G	V/G
Numeric	995 000 000	2 751 105	202 362	13.59
String	1 900 000 000	535 526	86 330	6.20
Reference	-	1 312 511	261 238	5.02
Date	101 000 000	3 437 420	379 240	9.06
Total	2 996 000 000	7 869 492	929 170	8.47

Table 6.5: Most frequent classes, properties and entities.

Class	Tables	Class	Property	Columns	Class	Entity	Rows
Company	88 729	BaseballPlayer	statisticValue	108 089	Company	Toshiba	59 112
Single	86 473	Single	musicalBand	27 781	Company	ShoreTel	45 600
PopulatedPlace	51 690	BaseballPlayer	activeYearsStartDate	25 596	Company	Nortel	45 573
BaseballPlayer	51 041	BaseballPlayer	activeYearsEndDate	21 412	Company	Avaya	45 562
Album	43 259	Single	album	19 970	Company	Mitel	45 509
Film	43 163	Album	releaseDate	17 944	Film	Georgia Georgia	24 337
FormulaOneRacer	36 696	VideoGame	releaseDate	17 468	Website	DVD Verdict	16 457
Country	36 009	Single	releaseDate	9 237	NascarDriver	Jeff Gordon	15 826
VideoGame	34 806	Film	releaseDate	8 250	NascarDriver	Kevin Harvick	15 110
NascarDriver	29 833	University	city	8 228	FormulaOneRacer	Fernando Alonso	14 870
SoccerClub	25 664	Single	musicalArtist	7 563	NascarDriver	Jimmie Johnson	14 826
SoccerPlayer	22 495	BaseballPlayer	number	7 554	NascarDriver	Tony Stewart	14 447
University	19 330	TelevisionStation	broadcastNetwork	6 849	NascarDriver	Matt Kenseth	13 815
Website	17 488	IceHockeyPlayer	birthDate	4 799	NascarDriver	Jeff Burton	13 744
AmericanFootb.Pl.	16 455	RugbyPlayer	birthDate	4 680	NascarDriver	Kyle Busch	13 539
City	16 169	MotorcycleRider	team	4 656	Country	China	13 515
TelevisionShow	13 891	VideoGame	publisher	4 636	FormulaOneRacer	Jenson Button	13 378
TelevisionStation	12 882	BaseballPlayer	birthDate	4 629	Country	France	13 300
IceHockeyPlayer	12 675	TelevisionShow	releaseDate	4 473	Company	IBM	13 271
Currency	12 251	FormulaOneRacer	championships	4 407	NascarDriver	Carl Edwards	13 218

6.4 Quality of Extracted Triples

This section analyses the quality of the triples that can be generated by fusing the values that are extracted from the matched web tables. The first subsection introduces the used methodology and compares three different approaches for data fusion. The second subsection then analyses the result of applying the data fusion to all created triples.

6.4.1 Methods

This section first introduces the methodology used for the large-scale evaluation of the extracted triples and for the evaluation of the quality of the triples that are selected by the data fusion methods. Based on this methodology, the quality of the extracted triples before fusion is discussed and three different data fusion approaches are introduced and evaluated.

Evaluation Methodology

To evaluate the data fusion results, the Local-Closed-World Assumption (LCWA) is applied [Dong et al., 2014a]. Under this assumption, the target knowledge base is considered complete for entity/property combinations for which it contains at least one triple. In the context of knowledge bases, usually the Open-World Assumption (OWA) is applied, which states that a statement may be true, even if it is not known. This means, using the OWA, it cannot be reasoned whether a newly discovered value is true or false if it is not contained in the knowledge base. In contrast, the Closed-World Assumption (CWA) states that only those statements are true which are also known to be true. Using the CWA, any newly discovered fact which is not contained in the knowledge base is considered false.

Although the CWA would allow an evaluation of the discovered facts, it would lead to an overestimation of false negatives, as every value that is missing from the knowledge base is considered false. Here, the LCWA offers a solution: if at least one value for a given entity/property combination is known, the knowledge base is considered locally complete and the CWA is applied; if no value is known, the OWA is applied. Formally, for all triples with subject s and predicate p , let $O(s, p)$ be the set of known objects in the knowledge base. If a newly discovered object o' is in $O(s, p)$, it is said to be correct; if o' is not in $O(s, p)$ and $O(s, p)$ is not the empty set, the new object is said to be incorrect; otherwise, no statement about the correctness is made.

As the triples that are extracted from web tables do not only contain entity references in their object position, but also literals, it is further necessary to extend the definition of the LCWA for literals. Other than entity references, literal values cannot be linked to existing entities in the knowledge base and checking literal values for equality can lead to many false negatives. For example, if one source states the population of Germany as 83 million, while the knowledge base contains the value

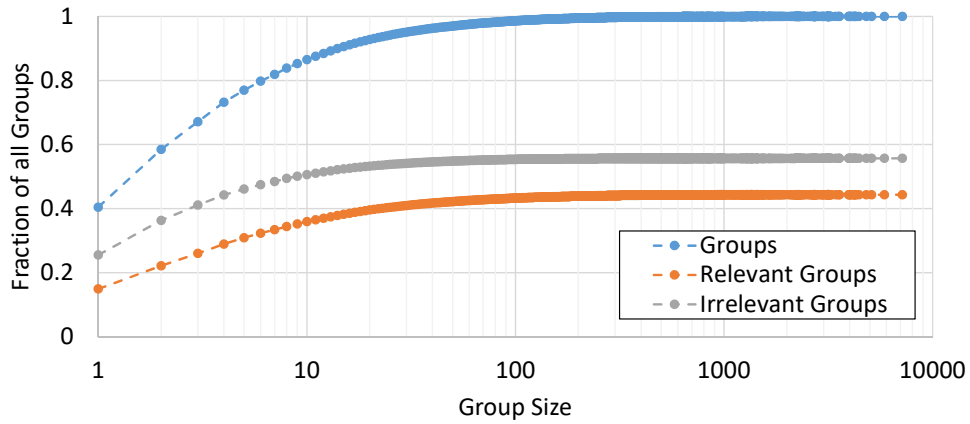


Figure 6.5: Distribution of relevant groups by size.

82.8 million, this should not be considered as incorrect. To establish whether a literal matches an existing value, a similarity measure can be used to allow for such small variations. Formally, a literal object value o' is considered to be in $O(s, p)$ if there exists an $o \in O(s, p)$ such that $\text{sim}(o, o') \geq \theta$. For `numeric` values, the similarity measure is the relative deviation $\text{sim}_{\text{numeric}}(a, b) = 1 - \frac{|a-b|}{\max(a,b)}$. For date values, the day, month, and year must match exactly if stated. For example, the date values 25-05-1978 and 1978 are considered a match. For `string` values, the Generalised Jaccard similarity with Levenshtein similarity for token comparison is used. Suitable thresholds are determined empirically from a manually annotated sample.

Relevant Groups for Data Fusion

The LCWA enables the large-scale evaluation of data fusion methods, but it also allows for the analysis of the extracted triples before fusion and the estimation of the amount of relevant groups for the data fusion task. Any data fusion method can only then select the correct value if it was extracted as a triple. If no triple in a group contains the correct value, a strategy can only choose to generate no value at all. Of the 929 170 groups that were created from the web tables, 691 622 have existing values in DBpedia, and 310 284 contain the correct value at least once. This means that 45% of all groups or 24.4% of all extracted triples contain a correct value. The fact that most of the extracted values are evaluated as false highlights how challenging the task of extracting data from web tables on a large scale is. This result is in agreement with state-of-the-art studies that report an accuracy of 30% [Dong et al., 2014b] and 11.5% [Dong et al., 2014a], respectively, for the extracted triples before fusion.

Table 6.6: Largest groups and evaluation according to the Local-Closed-World Assumption.

Data Type	Class	Property	Instance	Group Size	Web Sites	Correct Values
Date	VideoGame	releaseDate	Assassin's Creed: Revelations	7 160	16	3 979
Date	VideoGame	releaseDate	LittleBigPlanet 2	5 874	16	7
Date	Album	releaseDate	Live 1.0	5 096	35	110
Date	VideoGame	releaseDate	Naruto Ninja Council 2	4 217	18	-
Date	VideoGame	releaseDate	Sonic Dash	4 053	34	-
Numeric	BaseballPlayer	statisticValue	Jim Abbott	4 800	22	31
Numeric	BaseballPlayer	statisticValue	Jose Barrios	4 136	16	54
Numeric	BaseballPlayer	statisticValue	Todd Belitz	3 422	16	294
Numeric	RugbyPlayer	height	Peter Wallace	978	2	-
Numeric	RugbyPlayer	height	Quentin MacDonald	978	2	-
Reference	TelevisionShow	network	The Biggest Loser (season 1)	2 695	1	2 695
Reference	MotorcycleRider	team	Karel Abraham	993	75	907
Reference	City	country	Frankfurt	853	3	853
Reference	AmericanFootballPlayer	debutTeam	Janoris Jenkins	833	2	-
Reference	AmericanFootballPlayer	debutTeam	William Green	833	2	-
String	AmericanFootballPlayer	draftRound	Chris Rainey	836	4	833
String	AmericanFootballPlayer	draftRound	William Green	835	3	2
String	AmericanFootballPlayer	draftRound	Jaye Howard	833	2	832
String	Politician	orderInOffice	Martin Caton	363	1	-
String	Politician	orderInOffice	Alan Meale	314	1	-

Figure 6.5 shows the cumulative frequency of all groups (series “*Groups*”), all groups that contain a correct value (series “*Relevant Groups*”), and all groups that do not contain a correct value (series “*Irrelevant Groups*”). The chart shows that 26% of all groups consist of only a single triple which does not have the correct value. But even with increasing group size, there are more groups without correct value than with. This shows that the data fusion task is not trivial and a fusion strategy must be able to distinguish between plausible and implausible values. It can hence be assumed that a majority voting strategy will not produce satisfying results. Table 6.6 shows examples of the largest groups with and without correct values for each data type.

With these findings, the precision and recall measures for evaluation can be defined using the number of relevant groups $groups_{relevant} = 310\,284$. As in information retrieval, precision is the fraction of returned results that are relevant, i.e., the fraction of fused values $fused_{total}$ that are correct ($fused_{correct}$):

$$\text{Precision} = \frac{fused_{correct}}{fused_{total}}$$

Recall is defined as the fraction of relevant groups for which the correct value is returned:

$$\text{Recall} = \frac{fused_{correct}}{groups_{relevant}}$$

Fusion Strategies

The goal of the data fusion step is to decide which of the extracted values in a group with the same entity/property combination is selected as output. This step is necessary because different web sites may provide conflicting values or errors introduced by the matcher create additional, incorrect values. To decide for a final value, the following three data fusion strategies are compared:

Majority/Median Fusion (MM). The first fusion strategy is a baseline method that does not consider source trustworthiness. It applies a majority voting to select the fused value from the group of candidate values. For the data types `string` and `reference`, this strategy selects the mode, i.e., the most frequent value. For the data types `numeric` and `date`, it selects the median value. This strategy decides for the final value only based on the frequency of sources that state the same value and makes the decision for each group of values independently of all other groups.

Knowledge-based Trust (KBT). An extension of the MM strategy by a trust score for each triple. Using this trust score, the voting is changed to a weighted vote for `string` and `reference`, and to a weighted median for `numeric` and `date`. The trust score is estimated individually for each web table column. It is the percentage of correct values in the column of all values that are known from the

knowledge base. To estimate these trust scores, a 5-fold cross validation is used. For comparing values from the web table with triples from the knowledge base, the same similarity measures as described in Section 6.4.1 are used. The strategy can further reject to produce a fused value if the trust score of the column is below a minimum threshold of .35. Such a knowledge-based trust fusion has been shown to be effective for knowledge fusion in the context of web data [Dong et al., 2015, Yin and Tan, 2011].

PageRank-based Trust (PR). Similar to the KBT strategy, this approach assigns a trust score to each triple. The trust score is the normalized PageRank [Page et al., 1999] of the website from which the web table was extracted. This means, this strategy assigns trust to web sites as a whole based on the hyperlink structure in the Web graph rather than based on the correctness of the known values. The intuition is that popular web sites, which receive many incoming links, are more trustworthy than less popular web sites. The PageRank scores are calculated on the host-level graph with 128 billion hyperlinks from the 2012 version of the CommonCrawl.²

Fusion Strategy Selection

This section presents the experimental evaluation of the fusion strategies introduced above, which will be used to select the fusion strategy that is applied to all extracted triples. Each strategy is applied to all groups and evaluated on all fused triples that have existing values in the knowledge base. The results are shown in Table 6.7: for each strategy, the number of produced triples with existing values in the knowledge base (column “*Existing*”), without existing values (column “*New*”), and its performance, measured by precision, recall, and F1-measure are shown. The performance for the KBT strategy was determined by a five-fold cross validation and is the macro average over all folds.

The baseline approach with majority/median fusion (strategy “*MM*”) cannot filter out groups without any correct values and produces many incorrect triples, which results in a low precision of 37%. The high recall of 82%, however, shows that it is a suitable approach for the groups that contain the correct value. The knowledge-based trust strategy (“*KBT*”) can filter out groups with low trust score and increases precision by 27 percentage points to 64%. Also, only few triples are missed compared to the baseline, as the recall of 79% is only slightly lower. The strategy using PageRank-based trust (“*PR*”) does not improve the result over the baseline strategy and the best result in terms of F1-measure is achieved without filtering. Based on these results, the knowledge-based trust strategy is chosen for further profiling.

²<http://webdatacommons.org/hyperlinkgraph/2012-08/download.html#toc4>

Table 6.7: Number of existing and new triples and evaluation results by fusion strategy.

Strategy	Existing	New	Precision	Recall	F1
MM	691 622	237 548	0.369	0.823	0.509
KBT	377 165	63 953	0.640	0.789	0.707
PR	691 622	237 548	0.365	0.814	0.504

Manual Verification. The evaluation methodology is verified manually on a random sample of fused triples created by the KBT strategy. First, a random sample of 1 000 fused triples with existing values in the knowledge base is selected. These triples are then manually annotated as either matching the value in the knowledge base or not. Three human annotators determine a precision of .716. The automated evaluation used the proposed methodology determines a precision of .678. For 95.8% of all triples it makes the same decision as the human annotators about whether a triple is correct or not. This result suggest that the methodology is valid and produces a reasonable approximation of the actual quality of the fusion result.

A second experiment shows how the quality measure determined from the triples with existing values can be interpreted for triples with no existing value in the knowledge base. A random sample of 500 triples without existing value in the knowledge base is selected and manually annotated. Two human annotators determine a precision of .624 on this sample. The micro-averaged precision for triples with existing values using the same properties as the sample is .639. This indicates that the quality measure determined from triples with existing values is also a reasonable approximation of the quality of triples without existing values.

6.4.2 Fusion Results

This sections presents detailed results of the data fusion process using the selected Knowledge-Based Trust fusion strategy. The data fusion result is discussed with respect to the same aspects as the previously presented topical profile, i.e., data types, classes, and properties, showing the potential for slot filling.

Data Types. Table 6.8 shows the fusion results by data type. The columns “*Existing*” and “*New*” show the number fused triples that already exist in the knowledge base and that are not yet covered by the knowledge base, respectively. The last three columns show the performance, which is evaluated for the triples that exist in the knowledge base using the LCWA. The highest precision is achieved for `string` values, the highest recall for the type `reference`. The recall for the `numeric` values of 46% is very low, which indicates that the trust scores for this data type are mostly low and many correct triples are filtered out. Reasons for this can be numerical properties that change over time, such as the population of

Table 6.8: Data fusion performance by data type.

Data Types	Existing	New	Precision	Recall	F1
Numeric	26 637	10 329	0.643	0.463	0.538
Date	171 653	23 301	0.627	0.806	0.705
String	34 260	14 285	0.756	0.823	0.788
Reference	144 615	16 038	0.629	0.874	0.731
Total	377 165	63 953	0.640	0.789	0.707

countries and cities or the weight of athletes. Another potential reason is the ambiguity of properties in the knowledge base: a frequently mapped numerical property is `statisticValue`. It is not clear which statistic this property refers to from the definitions in the knowledge base and as a consequence, various columns with different semantics are mapped to this property, leading to low trust scores.

Classes. Table 6.9 shows detailed fusion results for the same set of classes shown in Table 6.3 earlier. The largest amount of new triples is created for the `Work` class, followed by triples for `Agent`. The highest precision is achieved for the classes `NaturalPlace` and `ArchitecturalStructure`. Aggregated to the top-level classes, `Place` and `Species` show a better performance with 80.0% and 84.5% F1-measure, respectively, than `Agent` (67.7%) and `Work` (70.4%). A possible explanation is that places and species have more functional or more static properties, i.e., properties that can only have a single value and/or do not change over time, such as `family` for `Species`, while works occur more frequently with multi-valued properties, such as `releaseDate`, which can differ depending on the region or edition.

Properties. Table 6.10 show the most frequent properties by number of new triples. The most frequent property is `releaseDate`, which is listed for `Film`, `Single`, and `Album`. In all three cases, the precision is rather low, which can possibly be explained by the fact that entities of these types can have different release dates for different regions or countries. With this argumentation, functional properties, i.e. those which can only have a single value, should have a higher precision. The shown properties support this assumption, as the highest precision is achieved for the `icaoLocationIdentifier` property for the type `Airport`.

Table 6.9: Data fusion performance by class.

Class	Fused Triples		Evaluation		
	Existing	New	Precision	Recall	F1
Agent	137 943	23 013	0.640	0.718	0.677
Person	117 365	15 001	0.639	0.718	0.676
Athlete	84 408	9 019	0.648	0.679	0.663
Artist	1 556	219	0.753	0.920	0.828
Politician	3 129	1 167	0.506	0.754	0.606
OfficeHolder	3 465	510	0.698	0.861	0.771
Organisation	20 522	7 903	0.647	0.716	0.680
EducationalInstitution	8 844	3 132	0.638	0.732	0.682
Company	6 376	2 547	0.706	0.851	0.772
Broadcaster	4 004	1 924	0.535	0.433	0.479
SportsTeam	790	132	0.659	0.901	0.761
Work	188 676	27 831	0.612	0.827	0.704
MusicalWork	118 482	8 419	0.597	0.827	0.693
Film	29 568	12 116	0.573	0.803	0.669
Software	17 554	2 766	0.593	0.763	0.667
Place	35 044	9 787	0.758	0.847	0.800
PopulatedPlace	15 500	6 508	0.713	0.788	0.749
Country	1 841	418	0.738	0.726	0.732
Settlement	531	224	0.580	0.722	0.644
Region	356	70	0.587	0.821	0.685
ArchitecturalStructure	11 934	2 352	0.806	0.940	0.868
NaturalPlace	743	64	0.874	0.970	0.919
Species	9 016	1 429	0.789	0.910	0.845
Total	377 165	63 953	0.640	0.789	0.707

Error Analysis. Based on a qualitative analysis through manual inspection of erroneous fusion results, the following types of errors can be identified:

- **Changes Over Time.** Values that are changing over time can lead to an incorrect evaluation with the applied evaluation methodology, as the value in the knowledge base and a value extracted from a web table might not refer to the same point in time. In such cases, a time-aware data fusion method could provide more accurate results [Oulabi et al., 2016].
- **Different Granularity.** Values can have different levels of granularity, e.g. the `city` property of “*Emroy University*” has the value “*Druid Hills, Georgia*” in DBpedia, which is a community in the metropolitan area of Atlanta. In the web tables, however, we find the value “*Atlanta*”, which is evaluated as false.
- **Missing Objects in Lists.** If a multi-valued property in DBpedia is incomplete, the automatic evaluation fails for cases in which a web table contains a correct, but missing value. This is by definition of the Local-Closed-World Assumption.
- **Conversion Issues.** Different formats for dates or numeric values can also lead to a false evaluation. For example, the `birthDate` of “*Jeff Zatkoff*” is “*6/9/1987*” according to DBpedia, but the data fusion returns the value “*9/6/1987*”. As web tables do not provide metadata that specifies the format, the date parser can misinterpret the data, in this case by confusing the day and month parts of the date.
- **Ambiguous Entities.** There is not always an equality correspondence between the entities in a web table and in DBpedia. For example, a web table might contain data about the release of a video game on a specific platform, such as the release date or the publisher, and these attributes can have different values depending on the platform or region in which the game is released. In DBpedia, however, all editions of the game are represented by a single entity and might only contain the first release date or publisher of the game.

Table 6.10: Examples of frequent properties after data fusion.

Class	Property	Type	Existing	New	Precision	Recall	F1
Film	releaseDate	Date	14 935	10 620	0.555	0.780	0.648
BaseballPlayer	number	Numeric	378	2 680	0.194	0.276	0.228
Single	releaseDate	Date	13 029	1 965	0.543	0.780	0.640
Book	publicationDate	Date	195	1 788	0.633	0.792	0.704
Album	releaseDate	Date	23 085	1 509	0.629	0.780	0.696
Company	locationCountry	Reference	463	1 192	0.677	0.900	0.772
Protein	symbol	String	58	926	0.660	0.660	0.660
RadioStation	licensee	String	275	910	0.788	0.983	0.875
WrittenWork	publicationDate	Date	48	894	0.574	0.792	0.666
Software	frequentlyUpdated	String	162	807	0.653	0.634	0.643
Person	birthDate	Date	14 216	782	0.545	0.897	0.678
PopulatedPlace	area	Numeric	94	705	0.639	0.660	0.650
Single	aSide	String	114	632	0.516	0.356	0.421
Bird	synonym	String	455	631	0.539	0.908	0.676
Airport	icaoLocationIdentifier	String	3 066	624	0.922	0.993	0.956
MusicalWork	artist	Reference	3 458	586	0.588	0.958	0.728
Person	deathDate	Date	5 204	574	0.671	0.914	0.774
Cyclist	alias	String	119	530	0.572	0.281	0.377
Person	status	String	8	515	0.200	0.355	0.256
PopulatedPlace	country	Reference	1 534	509	0.769	0.943	0.847

6.5 Conclusion

This chapter presented a topical profile for the Web Data Commons Web Tables Corpus 2012 with respect to the DBpedia knowledge base. First, related work in the areas of Web data profiling and Web data fusion was introduced in Section 6.2. Then, Section 6.3 presented the topical profile for web tables by analysing the created correspondences to classes, entities, and properties in the knowledge base. The presented profile is the most fine-grained topical profile of a corpus of web tables that is published at the time of writing. Finally, the quality of the extracted triples was analysed as an additional profiling dimension and several data fusion methods were compared in Section 6.4.

This chapter made the following contributions:

- **Topical Profile:** An in-depth analysis of the correspondences created between web tables and the DBpedia knowledge base provides a topical profile of the web table corpus that exceeds to scope of earlier studies and provides detailed insights into its contents. The results confirm that the distribution of class correspondences is correlated with the frequency of entities for these classes in the knowledge base. It further shows that most entities and properties are only found in few sources, while a small number of very popular entities and properties occur very frequently. In total, 949 970 web tables could be assigned a class from DBpedia. Further, 562 445 property annotations that represent relations between the subject column and another column in the web table, as well as 13 726 582 entity annotations for the values in the subject columns were assigned. At the time of writing, this is the most fine-grained topical profile of a corpus of web tables that is published.
- **Data Type Profile:** The analysis of the data type distribution for the matched columns reveals that the majority of the 7.9 million extractable triples contain numerical (2.8 million) and date (3.4 million) values. This is not considered by most state-of-the-art approaches for semantic table interpretation, which often focus exclusively on named entity columns.
- **Quality of extracted Triples** A heuristic estimation of the quality of the extracted raw triples and their inter-source distributions shows that many statements are only found in a single source and confirms that the overall extraction quality is low with the current state-of-the-art methods. Only 45% of all value groups by subject entity and property combination, or 24% of all extracted triples, have a correct value. A large number of high-quality facts can be extracted by applying knowledge-based trust data fusion, which results in 441 thousand fused triples with an estimated F1-measure of 0.7.

This chapter studied the contents of a large, publicly available corpus of web tables on an unprecedented scale and level of detail. Further, this is the first fully reproducible study in this area, as all data and implementations are openly available. The results of the profiling show that the majority of the web tables does not

contain data that can be related to the DBpedia knowledge base. Only 1.3% of all web tables that were extracted from the web crawl contained relational data. Of these relational tables, 3% could be matched to DBpedia. These percentages are comparable to the results of similar studies [Cafarella et al., 2008b, Venetis et al., 2011, Dong et al., 2014a, Hassanzadeh et al., 2015].

For the web tables that could be matched to DBpedia, the distribution of correspondences over entities and classes was found to be similar to the distribution of entities in DBpedia. While some deviations indicate over- and under-represented topics in the web table corpus, this result confirms the findings of Hassanzadeh et al. [Hassanzadeh et al., 2015], which indicate that the correspondence distribution depends on the content of the knowledge base. The statistics further revealed that properties could only be mapped for one third of the web tables, for which matching classes and entities were found in the knowledge base. This indicates a large potential for the schema-extension task, as the majority of the columns in web tables cannot be understood with the existing properties in DBpedia.

The frequency distributions of individual entities and properties reveal that 70% of the matching DBpedia entities are described in at least two Web tables. For properties this holds for 88%. However, the combination of both obtained by grouping the extracted triples by instance and property shows less replication of statements, as only 58% of these groups can be found in two or more sources.

By validating and applying a heuristic based on the Local-Closed-World Assumption, it has been shown that 45% of the created factual groups, or 24% of the extracted triples, contain a correct value. This, again, is in line with other studies that reported an accuracy of 30% [Dong et al., 2014b] and 11.5% [Dong et al., 2014a] for the extracted triples before data fusion. This shows that state-of-the-art semantic table interpretation methods still need to be improved for the application on a large scale.

Finally, the experimental evaluation and application of data fusion methods showed which types of facts and level of quality can be obtained with current methods. By analysing the results, several categories of systematic errors were identified, which provide directions for future work. With an overall F1-measure of 0.7, the fused triples are of high quality compared to the low quality of the raw extracted triples. For knowledge base augmentation with very high quality requirements, such as for productive systems, a human verification of the fused triples is, however, necessary. In such a verification process, the calculated trust scores can be used to suggest the values which are most likely correct to an end user, who then makes the final decision whether to add them to the knowledge base or not.

Chapter 7

Table Stitching

7.1 Introduction

The results of the corpus profiling in Chapter 6 indicate that the quality of state-of-the-art semantic table interpretation methods is over-estimated by the currently used evaluation datasets, which leads to low quality of the triples which can be extracted from web tables. This raises questions about the validity of existing evaluations and the actual performance of the evaluated methods. The data profile of the web table corpora presented in Chapter 4 shows that the vast majority of web table is very small, with a median of only 6 rows. This is not reflected in the selection of tables for evaluation datasets such as the Limaye gold standard, with a median of 21 rows, or the instance-level subset of the T2D gold standard, with a median of 100 rows. However, very small web tables pose significant challenges to semantic table interpretation methods, which rely on the presence of a sufficient amount of known entities in the web tables for their similarity calculation.

This chapter addresses the challenges that arise for very small web tables. The proposed method exploits the fact that most web sites are generated by templates and hence contain many web pages with the same structure. A profile of the frequency of web tables which use the same schema reveals that the same holds for the web tables on such web sites, i.e., the majority of the web tables in the corpus are generated from a template and hence use the same schema. Such web tables can be stitched (merged) to form larger tables, which can then be processed with high quality by existing semantic table interpretation methods. The experimental evaluation on a random sample of web tables shows that existing methods, T2K Match and COMA [Aumüller et al., 2005], fail to produce an acceptable result, especially for very small web tables. After the application of the proposed method, their performance reaches the level of quality that is expected from the existing evaluations, showing that the stitching procedure is effective and necessary to achieve high-quality results.




Table on main page				
Pos	Game	Weeks	Yearly	Total
1	 <i>Horizon: Zero Dawn (PS4)</i> Sony Computer Entertainment, Action	13	2,914,287	2,914,287
2	 <i>The Legend of Zelda: Breath of the Wild (NS)</i> Nintendo, Action	13	2,910,880	2,910,880
3	 <i>Resident Evil VII: Biohazard (PS4)</i> Capcom, Action	18	2,043,611	2,043,611

Table on Entity Page 1			
Title	Publisher	Region	Date
Horizon: Zero Dawn	Sony Computer Entertainment	North America	28th February 2017
Horizon: Zero Dawn	Sony Computer Entertainment	Europe	01st March 2017
Horizon: Zero Dawn	Sony Computer Entertainment	Japan	02nd March 2017

Table on Entity Page 2			
Title	Publisher	Region	Date
The Legend of Zelda: Breath of the Wild	Nintendo	Japan	03rd March 2017
The Legend of Zelda: Breath of the Wild	Nintendo	North America	03rd March 2017
The Legend of Zelda: Breath of the Wild	Nintendo	Europe	03rd March 2017

Table on Entity Page 3			
Title	Publisher	Region	Date
Resident Evil VII: Biohazard	Capcom	Europe	24th January 2017
Resident Evil VII: Biohazard	Capcom	North America	24th January 2017
Resident Evil VII: Biohazard	Capcom	Japan	26th January 2017

Figure 7.1: Example how information about video games is broken up into multiple small web tables on different pages.

Figure 7.1 shows examples of small web tables describing video games. It shows how information about the games that likely resides in a single table in the database behind the website is published as a set of small tables on different pages. From the example, it is easy to see why state-of-the-art algorithms fail: (1) there is only one main entity described in the table, but the commonly used subject column detection methods focus on uniqueness and will choose the wrong column, and (2) even if the correct entity label is detected, there is only one example from which the table must be mapped correctly, i.e., if the name of this single entity is not known, the table cannot be understood. However, the many small tables with the same schema can be combined into a single table, which solves the mentioned problems as the uniqueness of the correct subject column increases and more examples are available as additional entity names are added.

The contributions of this chapter are:

- **Re-Evaluation of Semantic Table Interpretation:** The existing evaluation datasets for semantic table interpretation are usually created by searching for web tables using seed entities from the knowledge base. This results in datasets that are biased towards larger tables and contain many of the entities in the knowledge base, which leads to over-estimated performance scores. A re-evaluation of the performance for semantic table interpretation on a random sample of web tables shows that the performance for small web tables is much worse than estimated by existing gold standards.

- **Data Profile of Schema Usage:** A data profile of the WTC 2015 shows that the majority of the schemata of web tables are re-used frequently by other web tables on the same web site. This indicates that most web tables are created by templates in the back-end of the web sites and can be combined to form larger tables.
- **Stitching Method:** Based on the profiling and evaluation results, a method is presented that stitches (combines) small web tables into larger tables based on their schema and holistic schema matching techniques. An evaluation of existing semantic table interpretation algorithms on a random sample of web tables shows that the quality of their results can be drastically improved by the stitching method, especially for web tables with only few rows and columns. The source code and all datasets that are used for the experiments in this chapter are publicly available.¹

This chapter is organised as follows. First, Section 7.2 introduces related work on merging web tables and holistic schema matching. Section 7.3 then introduces the proposed stitching method, provides a data profile of the schema frequencies in the web table corpus and presents the used schema matching techniques. The experimental evaluation of the proposed method and its impact on semantic table interpretation methods is presented in Section 7.4.

The work presented in this chapter, i.e., table stitching, has previously been published in [Lehmberg and Bizer, 2017].

7.2 Related Work

This section introduces the related work for table stitching. First, approaches that combine individual web tables or parts of these web tables are described and similar ideas to the method that is introduced in this chapter are discussed. Then, approaches for holistic schema matching, which allow the combination of web tables with different schemata are presented.

Merging Web Tables

The idea of stitching web tables from the same web site was originally proposed by Ling et al. [Ling et al., 2013] in 2013, but has not received much attention by the research community until four years later, when the method presented in this chapter [Lehmberg and Bizer, 2017] and a similar method focused on mapping relationships [Wang and He, 2017] were published. Most other studies concerned with web tables consider each web table as an individual data source and do not distinguish based on the web sites from which the web tables were extracted.

¹<https://github.com/olehmberg/WebTableStitching>

Matching web tables to each other and not to a knowledge base for the purpose of merging their data is mainly considered in the context of table extension, where the user provides a query table that is enriched with the data from matching web tables from a large corpus [Cafarella et al., 2009, Das Sarma et al., 2012, Yakout et al., 2012, Lehmborg et al., 2015]. The following paragraphs discuss some of these methods and highlight their differences to the method proposed in this chapter.

[Tao and Embley, 2009] Tao and Embley propose to use the template-based structure of web pages and detect so-called “*sibling tables*”. These tables are supposedly created from the same database and similar to each other. They use the simple tree matching algorithm [Yang, 1991] to match web tables from pairs of web pages to each other, and empirically define similarity thresholds to distinguish between identical tables, sibling tables, and unrelated tables. The detected sibling tables are then used to infer the table structure based on three pre-defined table layouts, which identify the positions of columns, headers, and values in the tables. The result of their method is a web-site-specific wrapper for the web tables that allows for their extraction and structural analysis from any web page with the same template, but they do not consider semantic table interpretation.

[Cafarella et al., 2009]. The OCTOPUS system proposed by Cafarella et al. provides a union operator that allows a user to merge tables. After searching related tables based on a query table provided by the user and possibly some manual pre-processing operations on the tables, the user can choose to create the union of selected web tables. This is a similar operations to that presented in this chapter, with the difference that it is supervised by an end user rather than being fully automated.

[Bronzi et al., 2013]. Bronzi et al. present a method that tackles both the wrapper induction and the schema integration problem for web pages that contain data about a single entity, called “*WEIR*”. The method is based on an “*abstract generative model*” that assumes that the data presented on each web page is based on a partial view over the same abstract relation. For each vertical, i.e., class, a single, complete relation is assumed, which is the source used by all web pages. The proposed model describes the generative process of the data providers, which select subsets of the attributes and tuples in the abstract relation and might introduce errors, imprecise values or missing values.

This model has two properties which are exploited by the proposed algorithm: “*local consistency*” and “*separable semantics*”. Local consistency states that attributes and their values are consistent among all web pages from the same data provider. Separable semantics states that all variations of an attribute that are published by different data providers are more similar to each other than to any other attribute.

These assumptions are similar to those made by the method presented in this chapter, i.e., that the data on different web pages from the web site are generated from a single data source and hence have a high consistency. However, the model by Bronzi et al. assumes a single abstract relation from which all web sites, theoretically, derive their data, while the method described in this chapter only assumes that the data on the same web site is generated from the same database.

[Ling et al., 2013]. Ling et al. argue that web tables from the same web site often use identical schemata because a large original table was split into multiple web tables for better human readability. They define web tables to be stitchable if their column headers are equal, regardless of their ordering, and produce union tables by concatenating all stitchable tables from the same web site. This approach is the foundation of the method that is presented in this chapter, which extends the set of web tables that are considered as stitchable.

[Wang and He, 2017]. Wang and He use the idea of table stitching as proposed by Ling et al. [Ling et al., 2013] and extend it to synthesize mapping relationships from web tables. A mapping relationship is a binary relation that maps, for example, country names to country codes. The proposed approach first identifies candidate column pairs in the web tables, then synthesizes two-column union tables, and finally resolves conflicts. They do not consider semantic table interpretation and evaluate the results of their method against a set of manually created mapping relationships. This evaluation measures the value overlap between the manually created relation and the best match in the generated corpus of mapping relationships, i.e., their proposed method is only concerned with generating the relationships, not with their semantic interpretation or possible methods for retrieving them from the corpus.

The literature presented in this section shows that the general idea of creating the union of web tables has been considered for a while, but only few approaches use it to improve the quality of methods that use web tables as data source. However, this changed in 2017 with the publication of two methods that use the idea of web table stitching for different use cases [Wang and He, 2017, Lehmberg and Bizer, 2017].

Holistic Schema Matching

Holistic schema matching refers to approaches that collectively match elements of a large corpus of schemata [Bernstein et al., 2011]. Methods in this area either avoid pair-wise comparisons completely or use the graph structure resulting from the correspondences that are created by traditional schema matching methods to improve the matching quality. Such methods are relevant for the approach presented in this chapter, which matches all web tables from the same web site among each other to find more stitchable tables.

[He and Chang, 2003]. He and Chang present a method for statistical schema matching. They propose to match input schemas holistically by finding an underlying hidden model from which all schemata in a specific domain can be generated. Such a model removes the need for pair-wise attribute correspondences, as it would explain all possible schemata for the domain. The proposed model consists of a vocabulary of attribute names and so-called “*concept partitions*”, which group semantically equal attribute names. For each attribute name and concept partition, the probabilities for being included in a schema that is generated from this model are determined. The estimation of the model parameters is based on three assumptions: (1) “*concept mutual independence*” states that different concepts are selected independently in schema generation; (2) “*synonym mutual exclusion*” states that no schema contains two synonyms for the same attribute; (3) “*non-overlapping concepts*” states that no two concept partitions share attribute names. Exploiting these assumptions in a schema matching method for web forms results in high quality results, and the schema matching method presented in this chapter also makes use of assumption (2) to improve the quality of the generated correspondences. Similar methods for complex schema mappings have been proposed later by He and Chang [He et al., 2004] as well as Su et al. [Su et al., 2006].

[Wang et al., 2004]. Wang et al. consider the problem of matching query interfaces on web pages to their respective result pages and to a global schema. The proposed method first generated correspondences between the elements of the query interface, result page, and the global schema using traditional schema matching methods. Then, the obtained schema mappings are checked for consistency by a method referred to as “*cross validation*”: A graph with attributes as vertices and correspondences as edges is created and initially partitioned by the mapping to the global schema. Inconsistent correspondences among the attributes are removed by minimizing the edge-cut of this partitioning as attributes are moved between partitions. A similar, graph-based refinement step is introduced in Section 7.3 of this chapter.

This section presented different approaches for holistic schema matching of web sources. Such approaches make use of assumptions about the data generation process to define constraints and use graph representations of the schema correspondences that are used to enforce consistency with a global schema. The method presented in the next section makes use of such constraints and a graph representation to improve the quality of schema matching among web tables.

7.3 Method: Table Stitching

This section introduces the idea of table stitching and defines the used terminology. First, the intuition behind table stitching is explained, and then the two stitching results, union tables and stitched union tables, which are created by table stitching are defined. The following subsections then describe the two stitching steps that produce these results in more detail.

Many web sites use content management systems or other programs to generate HTML pages. The web tables on these pages are often also automatically generated and the same schema is thus used by web tables on many different pages. Common practices are, for example, *paging* and *master-detail views*. Paging means that a long table is split into many smaller tables that are easier to understand for a human user, who can navigate between the pages if she is interested in seeing more data rows. Master-detail views consist of an overview table listing the most important attributes for all entities, and a detailed page for each entity that lists additional data. A user can then select the entity she is interested in from the master view and navigate to the detail view for this entity to access more information.

As a result of applying such practices, web tables tend to be small, which can be problematic for methods that try to infer the meaning of a table based on its content. The goal of *table stitching* is to merge the web tables that are created by practices such as paging or master-detail views into larger tables.

Formally, each web table can be considered as a view $v_i \in V$ on a relation instance r that applies a selection (σ) operation $v_i = \sigma_i(r)$. As proposed by Ling et al. [Ling et al., 2013], such web tables can be stitched into a union table r' by applying the union operation:

$$\text{Union Table: } r' = \bigcup_{v_i \in V} v_i$$

Definition 19 (Table Stitching) *Given a set of web tables \mathcal{W} which are the result of queries to an unknown relation instance r , reconstruct a relation instance r' that is consistent with the observations in \mathcal{W} . r' is consistent with \mathcal{W} if and only if there is a query q_i that produces w_i from r' for all $w_i \in \mathcal{W}$.*

Definition 20 (Union Table) *The result of table stitching under consideration of selection queries is called a union table.*

The original work of Ling et al. describes how web tables from the same web site can be stitched in order to create larger tables that are useful for visualising and mining the data. The authors define web tables to be stitchable if all their column headers match and create the union of all such tables from the same web site. However, the main focus of their work is to extract additional attributes from the web pages (i.e., from the URL, title, and text) on which the web tables were found. The work presented in this chapter has a different focus and investigates the effect of this procedure on semantic table interpretation.

In addition to web tables with exactly matching schemata, web tables with different, but overlapping sets of attributes, which are the result of different projections and possibly renamed attributes are also considered. In this case, each web table can be interpreted as a view $v_i \in V'$ on a relation instance r that applies selection (σ), projection (π), and renaming (ρ) operations $v_i = \rho_i(\pi_i(\sigma_i(r)))$. These web tables can be stitched into a larger table by matching their attributes, i.e., inverting the renaming $v'_i = \rho_i^{-1}(v_i)$, extending their schema such that all web tables use the same set of attributes (where missing attributes are filled with null values) and then creating their union. The last two steps correspond to applying the outer union operator (see Chapter 2), resulting in a stitched union table r'' :

$$\text{Stitched Union Table: } r'' = \bigcup_{v_i \in V'} \rho_i^{-1}(v_i)$$

Definition 21 (Stitched Union Table) *The result of table stitching under consideration of selection, projection, and renaming queries is called a stitched union table.*

It is reasonable to expect that the web tables provided by a single web site are generated from the same database [Ling et al., 2013]. However, not all web tables must be created from the same relation in such a database. In the case of stitched union tables, it must hence be decided which tables should be merged. One possibility is merge all web tables that have at least one common attribute. But, in the worst case this creates a single stitched union table with very low overlap between the original web tables. For the experiments in this chapter, stitched union tables only merge tables that have a common candidate key. With respect to the designated use case, improving the quality of semantic table interpretation, these candidate keys must further contain at least one string column, which can be used as subject column. All correspondences among web tables which do not fulfil this condition are ignored during stitching.

7.3.1 Union Tables

This section discusses the stitching of web tables with exactly matching schemata into union tables. The creation of union tables only requires the detection of attribute names in the web tables (see Chapter 4), so its impact can be analysed directly from the statistics of the used web tables corpus. For this analysis, the ordered set of column headers and the host part of a web table's URL, which is used to identify web sites, are considered as the schema of a web table (so no two schemata from different hosts are considered equal). Note that two schemata which are semantically, but not literally equal, for example when using different languages for their column headers, are considered as two distinct schemata. Such schemata are only stitchable for the creation of stitched union tables, but not for the creation of union tables.

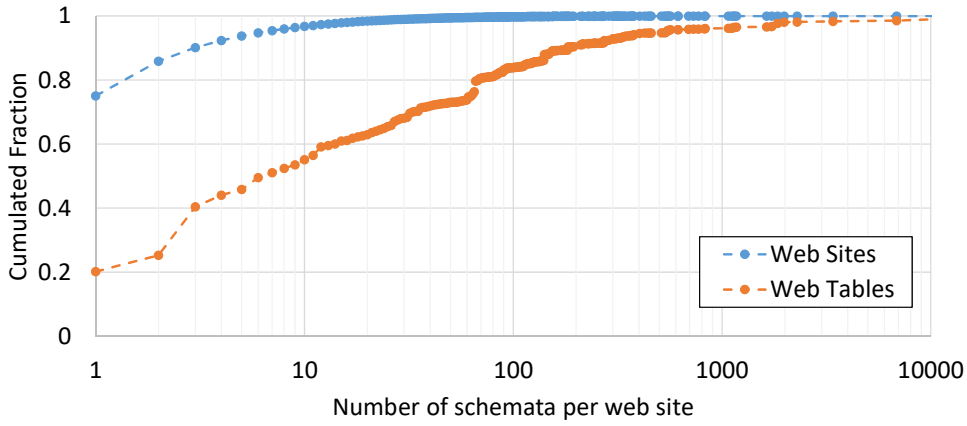


Figure 7.2: Number of schemata per web site.

Schema Profiling

This section presents a data profile for the schemata in the used web table corpus, which shows the potential for table stitching and supports the assumption that most web tables are generated by templates. This profile analyses the number of different schemata used by each web site as well as the number of web tables per web site that re-use the same schema.

Schemata per Web Site. Figure 7.2 shows the cumulated fraction of all web sites in the corpus that provide a certain number of different schemata. The series “*Web Sites*” shows the cumulated fraction of web sites which provide the number of schemata indicated on the horizontal axis. The series “*Web Tables*” shows the cumulated percentage of web tables from these web sites in the corpus. The chart shows that 75% of all web sites only use a single schema and that they provide 20% of all web tables. Further, 55% of all web tables are provided by web sites with up to 10 schemata (97% of all web sites) and 84% of all web tables by web sites with up to 100 schemata (99.7% of all web sites). This means that the majority of web tables are provided by web sites which use a small number of different schemata. This indicates that most web tables are created systematically and can be stitched into larger union tables.

Web Tables per Schema. Figure 7.3 shows the cumulated fraction of schemata which are used by a certain number of web tables. The series “*schemata*” shows the cumulated fraction of schemata which are used the number of web tables indicated on the horizontal axis. The series “*Web Tables*” shows the cumulated percentage of web tables that use these schemata. This shows that 63.7% of all schemata are only used once, but they only contribute 2.5% to the total amount of tables in the corpus. Even considering all schemata which are used up to 100 times (98.7% of all schemata) only cumulates to 17% of all tables. The remaining 83%

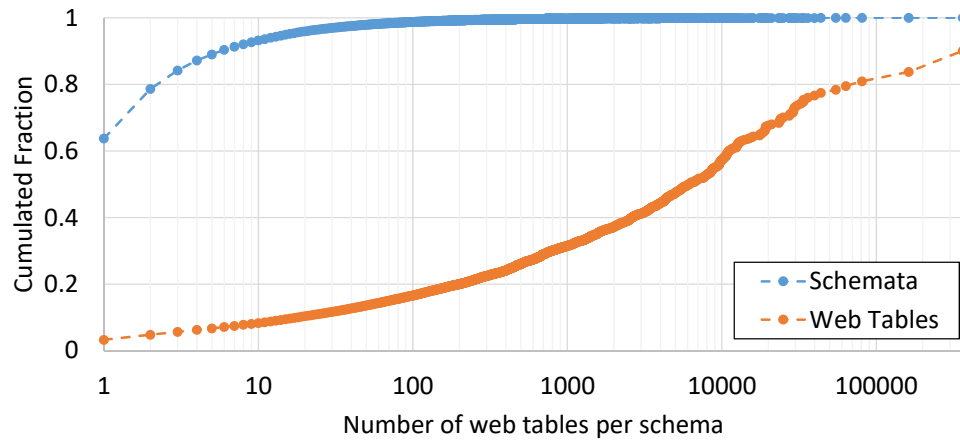


Figure 7.3: Number of web tables per schema.

of all tables have a schema that is used by at least 100 other tables on the same web site. This shows that for the majority of all web tables, there is a large number of other web tables with the same schema provided by the same web site, which can be stitched into union tables.

These results support the hypothesis that most web tables in the corpus are generated by content management or related systems. The majority of web tables is found on web sites that use a relatively small number of different schemata and most web tables re-use the same schema.

Assumption Violations

Web sites which provide a large number of different schemata likely violate the assumptions that are made by the stitching method. To understand the reasons for such a large number of different schemata, a manual inspection of the 20 web sites with the highest diversity of schemata is performed.

Together, these 20 web sites provide 60 942 schemata and 245 348 web tables. The analysis reveals that 20% of these web sites actually provide vertical tables (although detected as horizontal tables) and hence a data value was interpreted as column header. Another 15% have no headers, but the first data row was mistaken as schema by the header detection method, and 25% use data values in the attribute names (i.e., names of sports teams or the date). This means that the high number of different schemata is incorrect and the web sites do only use a much smaller number of schemata. Only 40% of the inspected web sites provide web tables with schemata that are manually created, i.e., the tables were not systematically created from a database. An example for this is the web site en.wikipedia.org, which clearly violates the assumption, and has 13 799 schemata in the used web table corpus, which are on average used by 3 web tables.



	Nome	Álbum	Duração	Preço
4	Wasted	Stabbing Westward	4:45	0,99 €

	Name	Album	Artist	Time	Price
1	Nasty Girl	Survivor	Destiny's Child	4:17	£0.99
2	Work (Freemasons...)	Work (Freemasons R	Kelly Rowland	3:11	£0.99
3	Until the End of Ti...	FutureSex/LoveSoun	Justin Timberlake	5:22	£0.99
4	Para Que Tu No Ll...	Vengo Venenoso	Antonio Carmona...	5:17	£0.99
5	Touch	Touch	Amerie	3:38	£0.99

Figure 7.4: Example of tables that are missed by the union approach.

7.3.2 Stitched Union Tables

This section discusses the stitching of tables with not exactly matching schemata. In this case, schema matching techniques must be used to find correspondences between the columns of the individual tables that should be stitched.

Examples of tables that cannot be stitched using the union approach are tables which either have the same schema but different column headers (for example, different languages or additional values in the column headers) or which have different, but overlapping schemata (for example a different set of attributes for the same entity). Figure 7.4 shows an example of two tables with stitchable schemata, which differ in the used language and the used set of attributes. The creation of stitched union tables requires the use of schema matching methods to determine which columns in the web tables correspond to the same attribute. In the following, the schema matching methods considered for this task are introduced.

Standard Matcher

The schema matching problem for union tables is different from the general web table matching problem, because it can be assumed that all data was generated from the same underlying table or database. This reduces heterogeneity and exact value comparisons can be used instead of similarity measures for comparing attribute values. To determine the performance of standard matching methods for this task, three matching approaches will be compared experimentally: label-based, value-based and duplicate-based schema matching.

Label-based Matcher. Label-based matching is implicitly applied when creating the union tables: all columns with the same header are matched to each other. This approach can be expected to also work for union tables which used different, but overlapping sets of attributes.

Value-based Matcher. In cases where column headers are missing or uninformative, it is reasonable to look at the values of the attributes. If two columns contain the same values, they might represent the same attribute. The value-based matcher measures the value overlap between two columns using exact value comparison. However, the weakness of this approach is that if two tables have different, but very similar attributes, for example “*birth date*” and “*founded date*”, it can be difficult to distinguish correct from coincidental matches.

Duplicate-based Matcher. If value-based similarities are misleading, duplicate-based schema matching is more promising. Given a set of duplicate records, this approach only compares the values of these duplicates in order to find schema correspondences [Bilke and Naumann, 2005]. For each pair of duplicate records, all values are compared, resulting in an attribute similarity matrix for each duplicate. These matrices are then aggregated by averaging, leading to a final attribute similarity matrix. With this approach, attributes with similar domains can be differentiated more precisely, as only attribute values from rows describing the same real-world entity are compared. In order to come up with a suitable set of duplicates, three different strategies are tested to estimate whether two rows in different tables refer to the same real-world entity: candidate keys, determinants, or subject columns.

Candidate keys uniquely identify a row and should hence result in an optimal set of duplicates. However, the experiments show that relying on candidate keys does not result in many duplicates. This can, for example, be explained by row numbers which are included in the web tables (see the first column in the web table in Figure 7.4) and are often included in the candidate keys that are discovered from the tables.

Determinants of functional dependencies are smaller sets of attributes than candidate keys and they do not necessarily uniquely identify rows in the table, which results in a larger number of duplicates. To determine functional dependencies and candidate keys the HyFD [Papenbrock and Naumann, 2016] functional dependency discovery algorithm is used.

Subject columns are commonly used in web table to knowledge base matching [Limaye et al., 2010, Mulwad et al., 2013, Venetis et al., 2011, Zhang, 2017, Ritze et al., 2015]. These columns contain the names of the entities that are described in a table and act as pseudo keys, resulting in the largest number of duplicates.

Hybrid Matcher

This section describes how the different standard matchers are combined into a hybrid matcher that combines their signals and how ideas from holistic schema matching are used to improve the consistency of the results.

The value- and duplicate-based matchers use different signals for their matching decisions than the label-based matcher, so they can be combined into a hybrid matcher. For this hybrid matcher, ideas from holistic schema matching that make use of the column headers, which are the strongest signal for this matching task, are included. Holistic schema matching refers to methods that make decisions based on observations from all schemata in a certain corpus [Yakout et al., 2012, He et al., 2004, He and Chang, 2003, Su et al., 2006].

Overall Process. Traditional matching methods only consider two tables at a time. But, if many tables have to be matched, pair-wise matching can produce inconsistencies. For example, two columns from the same table could be matched to each other via a path of correspondences to other tables. Such inconsistencies can only be detected when considering the mapping of more than two tables at the same time. The proposed hybrid matcher first uses one of the standard matchers from the previous section to perform a pair-wise matching between all union tables. After the pair-wise matching, two holistic refinement steps are applied to remove inconsistent correspondences and add missing correspondences to the mapping.

Pair-wise Refinement. The first refinement operates on pairs of tables and the schema correspondences that were created by the standard matcher. First, correspondences which are inconsistent with respect to all schemata on the same web site are removed as follows: Under the assumption that attributes are not duplicated in a single schema, the co-occurrence of attribute names in the same schema is used as negative evidence attribute equality. All column headers that appear together in at least one schema hence cannot be mapped to one another and such correspondences are removed.² This has been shown to be applicable for web tables [He et al., 2016] as well as the schemata of query forms on web pages [He and Chang, 2003]. To exclude violations of this assumption, all “*horizontally stacked*” tables as described in Section 8.4 are removed. Such tables are constructs where one schema is repeated multiple times in a single table to stretch it horizontally (for purely visual reasons). Second, correspondences for all columns with equal headers are added if they do not exist already, which is done by running the label-based matcher.

Graph-based Refinement. The second refinement operates on the graph of all schema correspondences as edges and columns as nodes. Again, the first step is to detect and remove inconsistencies. The same rule as during the pair-wise refinement is applied, but this time to the full graph. The correspondences in the graph are inconsistent if two columns with headers that co-occurred in a schema are connected by a path of schema correspondences. In such a case, the edge

²But if an attribute name occurs multiple times in a single schema, this does not lead to the removal of any correspondences.

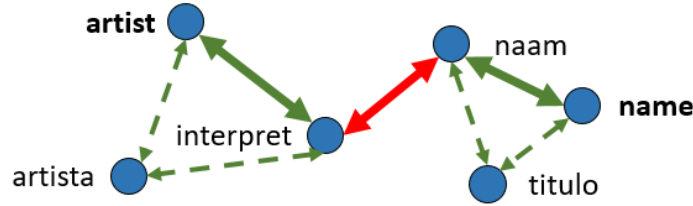


Figure 7.5: Graph-based refinement: If “*artist*” and “*name*” appeared in the same table, the path between these nodes (solid edges) can be detected as inconsistent. To solve this inconsistency, the red edge is removed, which has the highest betweenness centrality of all edges on this path.

with the highest betweenness centrality [Freeman, 1977] is removed from the inconsistent path. As an example, assume a perfect mapping between columns that represent two attributes. The resulting graph contains two connected components of columns, one for each attribute. Now an incorrect correspondence is added to this graph, which connects the two components, shown as the red edge in Figure 7.5. This new edge is part of all shortest paths that connect any two columns from the different components. So, in an inconsistent matching graph, it is desirable to remove the edge(s) that connect different components, which can be measured using betweenness centrality. The betweenness centrality measures the fraction of all shortest paths between any two nodes in the graph that include a certain edge. In the example, the inserted edge has the highest betweenness centrality value because it is the only connection between the two components. This procedure is similar to the one proposed by Wang et al. [Wang et al., 2004]. As final step of the graph-based refinement, all edges that can be inferred by transitivity are added to the graph, which reduces its incompleteness.

7.4 Improving Semantic Table Interpretation

This section presents the experimental evaluation of the proposed stitching method. First, the different schema matching methods for table stitching are evaluated. Then, the experiments evaluating the improvement for semantic table interpretation are presented.

Web Tables Corpus. The following experiments use a subset of the Web Data Commons Web Tables Corpus 2015. This subset contains 5 176 160 de-duplicated web tables from the corpus in which at least one entity from the DBpedia knowledge base could be recognised. Tables without any recognisable entity from the knowledge base cannot contribute to the knowledge base augmentation task and can hence be excluded. The detection of entities was performed by comparing the

table's tokenised cell values to the entity labels in DBpedia with Jaccard similarity and a threshold of 0.7. The tables in the subset have a median of 6 rows and originate from a total of 86 316 different web sites.

Union Tables. Creating the union tables for all web sites in the corpus reduces the total number of tables from 5 176 160 to 261 215. The average size of the tables increases from 9 rows to 108 rows. During the creation of the union tables, duplicate rows are removed, which removes on average 8% of the rows from the original web tables.

To evaluate the hypothesis that table stitching improves the quality of semantic table interpretation methods, the results of two matching systems, T2K Match and COMA, are compared for web tables, union tables, and stitched union tables.

T2K Match. T2K Match (see Chapter 5) is used as an example of a specialised web table to knowledge base matching system in the experiments. T2K Match matches web tables to classes, table rows to entities and columns to properties in a target knowledge base. Before matching a web table, one column is determined to be the subject column, i.e., the column that contains the names of the entities that are described in the table. The detection of the subject column is performed heuristically, by choosing the most unique text column. Afterwards, candidate correspondences to entities and properties from the knowledge base are used to determine the majority class of the entities in the web table. Then, entity and property correspondences are iteratively refined, until a final mapping is created.

COMA. To test the general applicability of the stitching method, the experiments are repeated using COMA 3.0 [Do and Rahm, 2002, Aumüller et al., 2005]. COMA is a general-purpose schema matching tool which exploits both attribute names and data values. As COMA is not tailored for the web table to knowledge base matching task, the experimental set-up is changed in the following way: The experiment is limited to the column-to-property schema matching task and COMA is run on the combination of each web table and its corresponding class in the knowledge base (so COMA does not have to find the correct class mapping). Due to limitations of the COMA application, a maximum of 1 000 data values per attribute limits the size of the used tables. Hence, for each matched table, 1 000 entities are sampled from the corresponding DBpedia classes. It is guaranteed that the entities corresponding to those in the web tables are included, then random entities are added until 1 000 rows are reached.

7.4.1 Intra-Site Schema Matching

In the following, the performance of the schema matching methods for the creation of stitched union tables is evaluated on several datasets. This schema matching task has the goal of matching all union tables created from the web tables of a single web site among each other, resulting in correspondences among their columns that can be used to create stitched union tables. After the evaluation of the performance of this intra-site schema matching, the improvement gained by table stitching for semantic table interpretation is evaluated on these datasets.

Datasets

To evaluate the proposed matching methods, several web sites with different characteristics are selected from the corpus. Table 7.1 gives an overview of these datasets. Each dataset contains all web tables from the respective web site (identified by the host part of its URL) that are included in the 5 million table corpus. The table shows how many web tables, rows and columns each dataset contains and the column “*Union*” shows the number of schemata, which is also the number of union tables that were created for the respective web site.

The goal of these experiments is to measure how correct and complete the results of the different schema matching approaches introduced in Section 7.3 are. For this purpose, a reference mapping is manually created for all datasets. These reference mappings are complete, i.e., they contain all correct correspondences between the columns of the union tables of the respective datasets.

Table 7.1: Datasets for schema matching of union tables.

Dataset	Tables	Rows	Columns	Union	Topic
data.bls.gov	10 825	54 196	59 093	10	statistics
itunes.apple.com	47 729	494 302	258 275	36	music
websiteviewer.com	11 351	99 865	39 535	4	statistics
www.nndb.com	23 522	231 738	116 716	9	multiple
seatgeek.com	157 581	2 644 035	630 063	64	multiple
vgchartz.com	23 258	58 637	116 285	6	video games
Total	274 266	3 582 773	1 219 967	129	

Standard Matcher Experiments

This section presents the results of the schema matching experiments for all matchers described in Section 7.3.2 and discusses their individual performance. An overview of these results is shown in Table 7.2.

Table 7.2: Comparison of all schema matchers (F1-measure, best configuration for each dataset is shown in boldface).

	bls	itunes	ws.looker	nndb	seatgeek	vgchartz	avg.
Standard Matchers							
Value	0.667	0.610	0.435	0.381	0.197	0.880	0.528
Subj. Col.	0.380	0.448	0.200	0.100	0.265	0.194	0.264
Det.	0.145	0.511	0.250	0.381	0.458	0.353	0.350
Key	0.031	0.147	0.000	0.111	0.165	0.194	0.108
Label	0.752	0.185	0.727	1.000	0.827	0.634	0.688
Hybrid Matchers							
Value	0.894	0.807	0.800	0.515	0.778	0.963	0.793
Subj. Col.	0.847	0.710	0.714	0.872	0.725	0.698	0.761
Det.	0.836	0.788	0.727	1.000	0.872	0.698	0.820
Key	0.844	0.528	0.727	1.000	0.842	0.634	0.763

Label-based Matcher. The label-based method is very strong in this use case and is overall the best standard matcher with an average F1-measure of 0.69. It achieves a precision of 100% for all datasets, with a recall varying between 10% and 100%. The *itunes* dataset contains tables in multiple languages, resulting in the worst performance for this matcher over all datasets with 0.19 F1-measure.

Value-based Matcher. The value-based matcher results in the second highest average F1-measure with 0.53. However, its precision is the lowest with an average of only 0.45. The problem is that seemingly similar columns are mapped to each other, which actually represent different attributes. For example, this results in mappings between “*artist*” and “*album*” for *itunes*, “*text*” and “*primary country*” for *websitelooker*, “*employment per thousand jobs*” and “*percent of state employment*” for *bls*, and between “*founded*” and “*birth date*” for *nndb*.

Duplicate-based Matcher. The three approaches for duplicate-based matching achieve the lowest average F1-measure values with 0.26 for entity-labels, 0.35 for determinants and 0.11 for candidate keys. These methods can only create correspondences if two tables contain records that are duplicates, so all correspondences between tables with distinct contents are missed. The precision of the duplicate-based methods is, however, comparably high with averages of 0.65 for subject columns, 0.89 for determinants, and 0.78 for candidate keys. When using subject columns to find duplicates, a problem is that ambiguous names cause incorrect duplicates, which in turn reduces the schema matching performance. This problem is reduced when using determinants or candidate keys.

Hybrid Matcher Experiments

This section presents the results of the hybrid matcher in configurations with all standard matchers. The hybrid matcher combines instance- and label-based schema matchers and adds a graph-based refinement step which is supposed to increase the consistency of the matching result.

Table 7.2 shows the F1-measure for all standard and hybrid matcher configurations (the best configuration for each dataset is shown in boldface) and Figure 7.6 shows the precision and recall achieved by all matchers for each dataset. In all configurations, the hybrid matcher outperforms all of the individual standard matchers averaged over all datasets. The best result is achieved by the determinant matcher, with an average F1-measure of 0.82.

A large improvement over the instance-based standard matchers can be attributed to the correspondences from the label-based matcher, because it can find the matches between tables with disjoint values. This increases the recall of the matchers by adding high-quality correspondences without introducing noise, as the label-based standard matcher has a precision of 100% on all datasets. In combination with the transitivity applied by the graph-based refinement, the value- or duplicate-based and the label-based correspondences complement each other. For example for the `itunes` dataset with the determinant-based matcher, this results in a recall of 0.69, which exceeds the combined recall of label-based (0.1) and determinant-based (0.39) matching.

Finally, the removal of inconsistent correspondences leads to an improvement in precision for all configurations and datasets and does not cause a decrease in recall. This effect is strongest for the configuration with the value-based standard matcher, which tends to generate too many correspondences that are effectively filtered out by the graph-based refinement step.

These results show that the combination of the different standard matchers into a holistic hybrid matcher improves the schema matching performance. Depending on the used standard matcher, a trade-off between precision and recall can be observed: While the value-based matcher still achieves the highest recall at comparably low precision, the duplicate-based methods achieve a higher precision and, in combination with determinants for the generation of duplicates, also the highest F1-measure.

7.4.2 Evaluation on Individual Web Sites

This section presents experiments for the evaluation of the improvement resulting from table stitching for semantic table interpretation. In these experiments, T2K Match is run on the stitched union tables created by the different configurations of the hybrid matcher on the datasets used in the preceding section. The evaluation will focus on the class and relation annotation task of semantic table interpretation (see Chapter 5), as the entity annotation task only considers individual rows and does not benefit from table stitching directly.

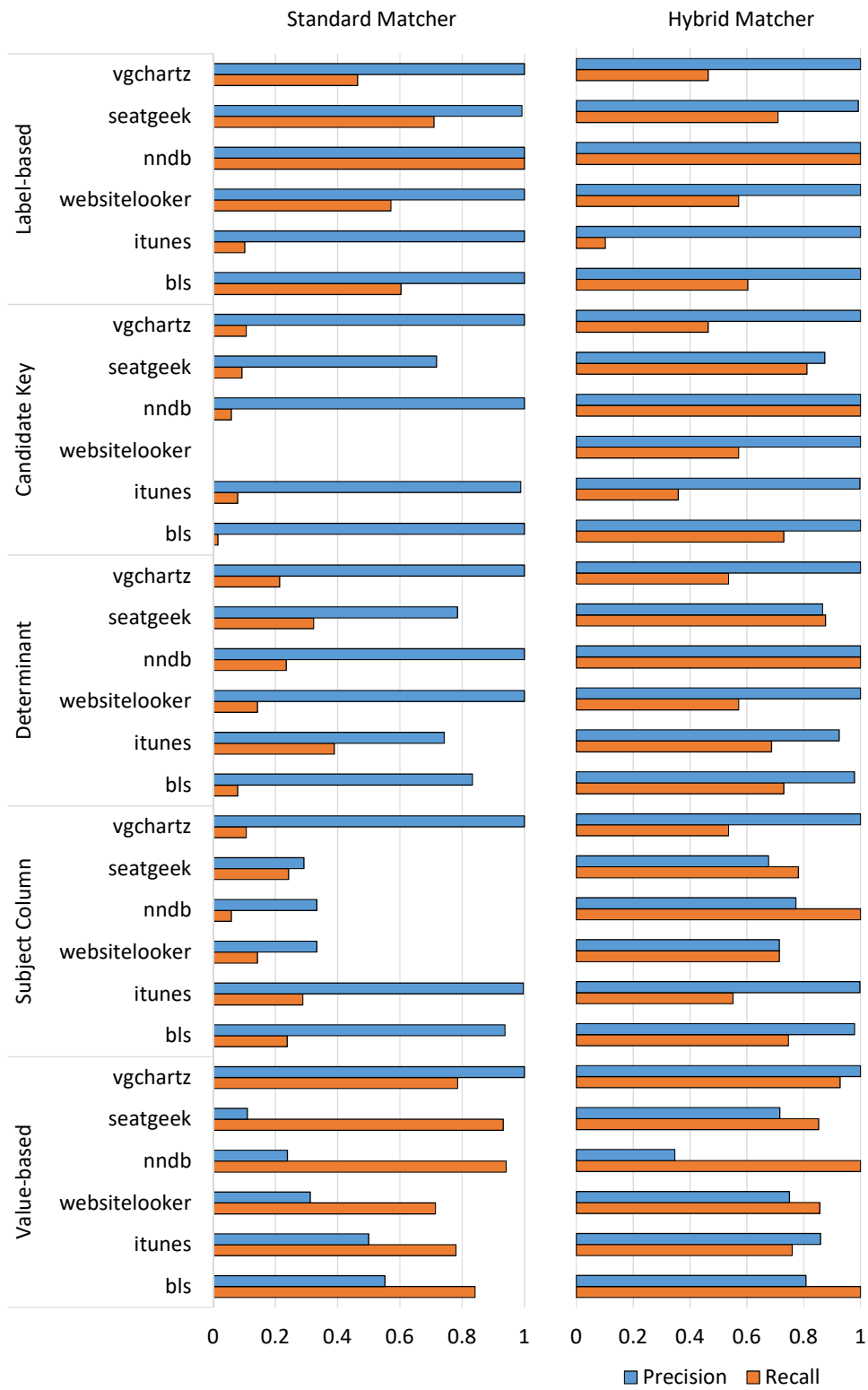


Figure 7.6: Precision and recall for all matcher configurations.

A precondition for the relation annotation task for semantic table interpretation is that the tables contain both entities and attributes which correspond to existing elements in the knowledge base. As the `websitelooker` and `bls` datasets do not contain any attributes that correspond to properties in the knowledge base, they are excluded from this experiment. The hybrid matcher is applied to all remaining datasets in its different configurations. Table 7.3 shows the number of tables and columns that result from the stitching with the different matchers. These results show that the number of tables and columns is drastically reduced for the `itunes` and `seatgeek` datasets compared to the union tables by further stitching them.

Table 7.3: Datasets for stitched union table to knowledge base matching.

		itunes	nndb	seatgeek	vgchartz
Union	Tables	36	10	64	6
	Columns	226	28	223	29
Value	Tables	1	7	2	2
	Columns	10	20	7	17
Subj. Col.	Tables	1	7	2	2
	Columns	12	23	7	18
Det.	Tables	1	9	4	3
	Columns	10	27	15	19
Key	Tables	1	9	5	3
	Columns	16	27	18	19

Table 7.4 shows the F1-measure resulting from running T2K Match on the original, union, and the stitched union tables for all configurations of the hybrid matcher. The reference mapping for all datasets was created by manually labelling the union tables with their corresponding properties in DBpedia and transferring the correspondences to all original web table columns. The performance measures are calculated based on the correspondences that are created for the original web tables in all configurations, i.e., for union tables and stitched union tables, all original web table columns which are stitched into a column that was annotated by T2K Match are annotated with the same property.

For the original tables, T2K Match manages to create rather precise results with low recall for the `itunes` and `nndb` datasets, but fails for the `seatgeek` and `vgchartz` datasets. The average F1-measure is 0.32. The tables in the `vgchartz` dataset can be extremely small and the result completely depends on whether the one entity in the table can be correctly recognised. For the `seatgeek` dataset, the described venues are not recognised as main entity, so no schema correspondences can be created.

With the union and stitched union tables, the average F1-measure increases to a range from 0.8 to 0.88. Most of the attributes which exist in the knowledge

base have been correctly matched by all stitching approaches, with the result that the performance of T2K Match does not differ much on these datasets. However, there are improvements for the `itunes` dataset, where the smaller union tables resulted in missed and incorrect correspondences compared to the larger stitched union tables. For these tables, the stitched union tables obtained with the determinant-based matcher resulted in the best performance of T2K Match. For the `vgchartz` dataset, the results of the value-based and subject column-based stitching approaches cause a worse performance of T2K Match compared to the union tables and the other stitching approaches. This is because union tables which only have a single attribute in common are matched, leading to wide tables with many missing values. With the other approaches, more matching attributes are required when finding duplicates and hence this error is avoided.

Table 7.4: Evaluation of T2K Match for web tables, union and stitched union tables.

	itunes	nndb	seatgeek	vgchartz	avg.
Web Tables	0.500	0.515	0.038	0.244	0.324
Union Tables	0.799	0.999	0.799	0.800	0.849
Value-Based	0.913	0.997	0.800	0.500	0.802
Subject Column	0.982	0.998	0.799	0.500	0.820
Determinant	0.944	0.999	0.800	0.800	0.886
Candidate Key	0.897	0.999	0.800	0.800	0.874
Label-based	0.930	0.999	0.800	0.800	0.882

Overall, there are improvements of 0.48 to 0.76 points in F1-measure with the union and stitched union tables in comparison to the original web tables. In all cases, creating the union tables causes a huge increase in F1-measure. Depending on the dataset, the additional stitching step after creating the union tables can further improve the results. However, as not all attributes can be mapped to the knowledge base, the differences between the stitching approaches are only slight compared to the results in the preceding section.

7.4.3 Evaluation on a Random Sample

This section presents experiments that evaluate the improvements by table stitching on a random sample of web tables. Existing gold standards for semantic table interpretation contain web tables that were selected using seed entities. This results in a selection of tables in which many entities from the knowledge base are found and which are generally larger to cover more entities. For example, the tables in the T2D gold standard have a median length of 100 rows and the tables in the Limaye gold standard, as re-built by Bhagavatula et al. [Bhagavatula et al., 2015]³, have a

³<http://websail-fe.cs.northwestern.edu/TabEL/>

median of 21 rows. The tables in the WTC 2015 (see Section 4.4.2), however, have a median of only 6 rows. This means that the gold standards are biased towards larger web tables.

To create an unbiased test set, a random sample of 1 000 web tables is drawn from the corpus described earlier. The sample contains web tables from 401 different web sites. In total, these web sites contribute 3.5 million web tables to the 5 million tables subset discussed above. The web tables in the 1 000 table sample have between 2 and 432 rows and a median of 4 rows. By manually annotating the tables, a reference schema mapping consisting of 427 column-to-property correspondences is obtained. Of these correspondences, 204 refer to the `rdfs:label` property (containing the names of the entities) and 223 to other DBpedia properties. This means that 204 web tables in the sample have at least some overlap with the knowledge base, while the remaining tables cannot be matched at all. The following sections will discuss the performance of T2K Match and COMA on this sample using the original web tables, union tables, and stitched union tables. Table 7.5 gives an overview of the results for the different experiments.

Table 7.5: Results of running T2K Match and COMA with web tables, union tables and stitched union tables.

T2K Match	Precision	Recall	F1
Web Tables	0.196	0.319	0.243
Union Tables	0.400	0.656	0.497
Value-based	0.578	0.590	0.584
Subject Column	0.563	0.562	0.563
Determinant	0.633	0.607	0.620
Candidate Key	0.623	0.604	0.614
COMA			
Web Tables	0.543	0.281	0.370
Union Tables	0.566	0.400	0.469
Stitched Union	0.617	0.433	0.509

Web Tables

T2K Match and COMA are run on the random sample of 1 000 web tables, resulting in a performance for the schema matching task that is much worse than on the T2D gold standard. On the gold standard, T2K Match achieves an F1-measure of 0.7 for the schema matching task. On the sampled tables, however, the achieved F1-measure is only 0.24. The achieved precision is 0.20 with a recall of 0.32 and 95 of the 204 map-able web tables are correctly identified. The reason for the low precision is that non map-able tables are mapped to the knowledge base, i.e., a correspondence to `rdfs:label` is created for 444 tables. Of these tables, 90%

have only six rows or less. For the 109 web tables that were not mapped although it would have been correct, there is also indication that the size of the tables could be a problem: 52% of these tables have only six rows or less. COMA achieves an F1-measure of 0.37 with a precision of 0.54 and a recall of 0.28. Other than T2K Match, COMA creates only 18 correct correspondences to the `rdfs:label` property, but 110 correct correspondences to other properties. It is noticeable that the correct correspondences are between columns and properties with similar or equal labels. This indicates that COMA did not consider the similarity of the data values to suffice in order to create correspondences in many cases and only mapped columns with matching headers. This baseline experiment shows that both systems do not achieve satisfying results for matching the sampled tables.

Union Tables

Next, T2K Match and COMA are run on the union tables. For the 401 web sites in the sample, which contain a total of 3.5 million web tables, 16 367 union tables are created. Each correspondence that is created for a column in a union table is transferred to all original columns in the web tables that were combined into this union table. Then, the correspondences for the same random sample of tables as in the previous experiment are evaluated.

On the original web tables, T2K Match achieves an F1-measure of 0.24. With the union tables, this is improved to an F1-measure of 0.5. There is an increase in the number of correct `rdfs:label` correspondences from 95 to 169. For all correspondences, a precision of 0.4 and a recall of 0.66 are achieved. The reason for the still rather low precision is that in total 727 tables were matched to the knowledge base, although only 204 of the sampled tables can actually be mapped. COMA achieves an F1-measure of 0.37 on the original web tables, which is increased to 0.47 for the union tables. The number of correct correspondences to `rdfs:label` is increased from 18 to 62 and overall a precision of 0.57 and a recall of 0.4 are achieved.

Stitched Union Tables

Finally, the stitched union tables are created and used to evaluate how this changes the schema matching performance. The stitching is run on all web sites in the random sample of 1 000. Again, each correspondence for a column in a stitched union table is transferred to all original web table columns that were merged into this column and all correspondences for the web tables in the initial sample are evaluated. The stitching procedure results in 1 160 stitched union tables for the value-based matcher, 1 562 for the entity-label matcher, 1 981 for the determinant matcher and 2 388 stitched union tables for the candidate-key matcher.

For T2K Match, the initial result of 0.24 F1-measure is improved to 0.50 when creating the union tables. With the stitched union tables, the best performance is achieved in the configuration with the determinant matcher, with an F1-measure

of 0.62. The additional improvements are due to an improved precision for correspondences to `rdfs:label` and improvements in both precision and recall for other properties.

For COMA, there is also a further improvement in both precision and recall. In this experiment, only the determinant-based matcher is used to create stitched union tables. The F1-measure increases from 0.37 for the original web tables and 0.47 for the union tables to 0.51 for the stitched union tables. Although the performance is worse than for T2K Match, which is specialised for this task, there is an improvement when using union tables (10%) and stitched union tables (14%) instead of the original web tables.

These experiments show that the step from original web tables to union tables results a strong increase in performance on the sample of 1000 web tables. It is thus recommended for any web table matching system to at least implement creating union tables as a pre-processing step, which does not require any schema matching. If the matching results should be further improved, systems can decide to combine union tables into stitched union tables using schema matching.

7.4.4 Result Analysis

In the previous section, it has been shown that stitching web tables can improve the results for semantic table interpretation. In this section, the impact of the stitching procedure on the amount of correctly extracted values and the amount of tables in the corpus is analysed. Afterwards, it is investigated which characteristics of web tables lead to the highest performance gains when using table stitching.

Amount of Extracted Values

With respect to applications that use the data from the web tables, such as the slot-filling knowledge base augmentation task, it is not only important to correctly match the schema, but rather how many values are extracted correctly.

Based on the schema and entity mapping created by T2K Match, this amount of values that can be correctly extracted is determined. Correctly extracted in this context means that the entity and property correspondence is correct, which is checked manually by verifying all correspondences that were created by T2K Match, but not that the actual attribute value is true. Compared to the results in Chapter 6, where 24.4% of all extracted triples contained a correct value, the evaluation on the random sample results in a precision of 39%. Figure 7.7 shows that by using union tables and stitched union tables, more values can be extracted at a higher precision than for the original web tables. The precision increases from 39% for original web tables to 55% for union tables and 59% for stitched union tables.

Note that the total amount of extracted values decreases for stitched union tables. As shown in Table 7.5, the second stitching step increases the precision of the matcher by 23.3 percentage points, but also reduces the recall by 4.9 percentage

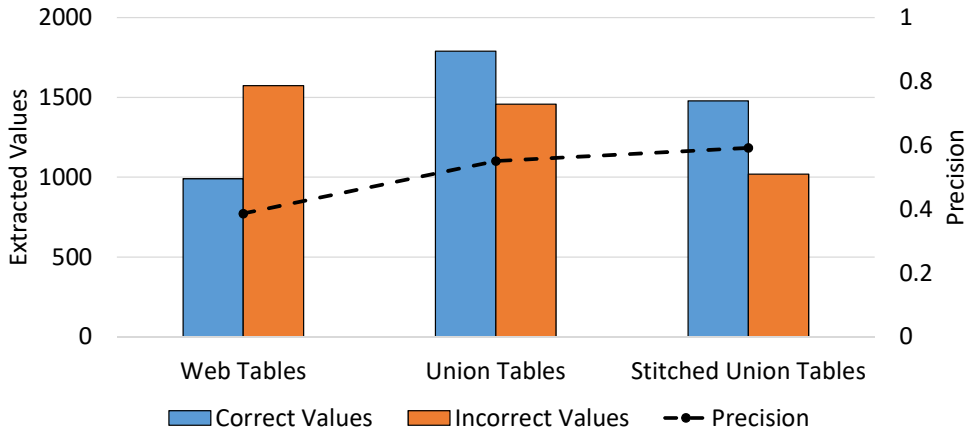


Figure 7.7: Evaluation of extracted values.

points, i.e., the values from fewer web tables are extracted. Closer inspection of the results shows that a few large web tables were correctly matched for the original web tables and the union tables, but not for the stitched union tables. This indicates that the schema matching among the union tables was incorrect and the stitched union tables hence contained inconsistent data that was rejected by T2K Match. As the majority of web tables is, very small, these few large tables have a large influence on the amount of extracted triples on the random sample.

Table Stitching Statistics

The result of the stitching procedure is the combination of large numbers of small web tables into larger tables. This reduces the overall amount of tables in the corpus as well as the amount of tables for individual web sites. Besides the demonstrated benefit for semantic table interpretation, a reduced amount of tables can also be beneficial for other applications. For example, table search [Cafarella et al., 2009, Yakout et al., 2012, Lehmborg et al., 2015] or interactive exploration of a table corpus [Ellis et al., 2015] are simplified by a reduced amount of tables.

In order to see the impact of table stitching on such applications, the number of tables per web site at each stitching step for the used web tables corpus is measured, shown in Figure 7.8. The horizontal axis shows the number of tables per web site, in terms of web tables, union tables, or stitched union tables depending on the series, and the vertical axis indicates the cumulated percentage of original web tables. For generating these statistics, the hybrid matcher in its configuration with determinant matching is used.

In the original corpus (series “*Web Tables*”), many web sites provide large amounts of web tables with a total of 5 176 160 web tables. 50% of these web tables are found on web sites with up to 23 464 tables. After creating the union tables, the total number of tables is drastically reduced to 261 215. Now, the data of 50% of all original tables is provided by web sites with no more than 7 union

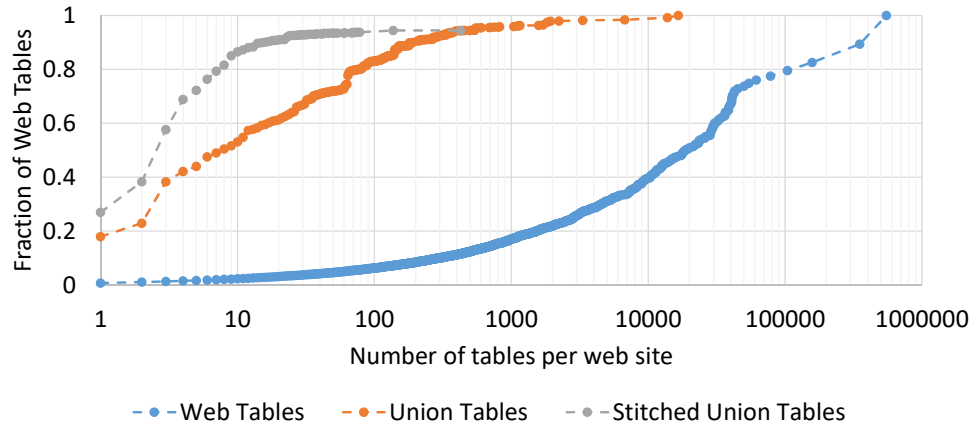


Figure 7.8: Tables per web site at different stitching steps.

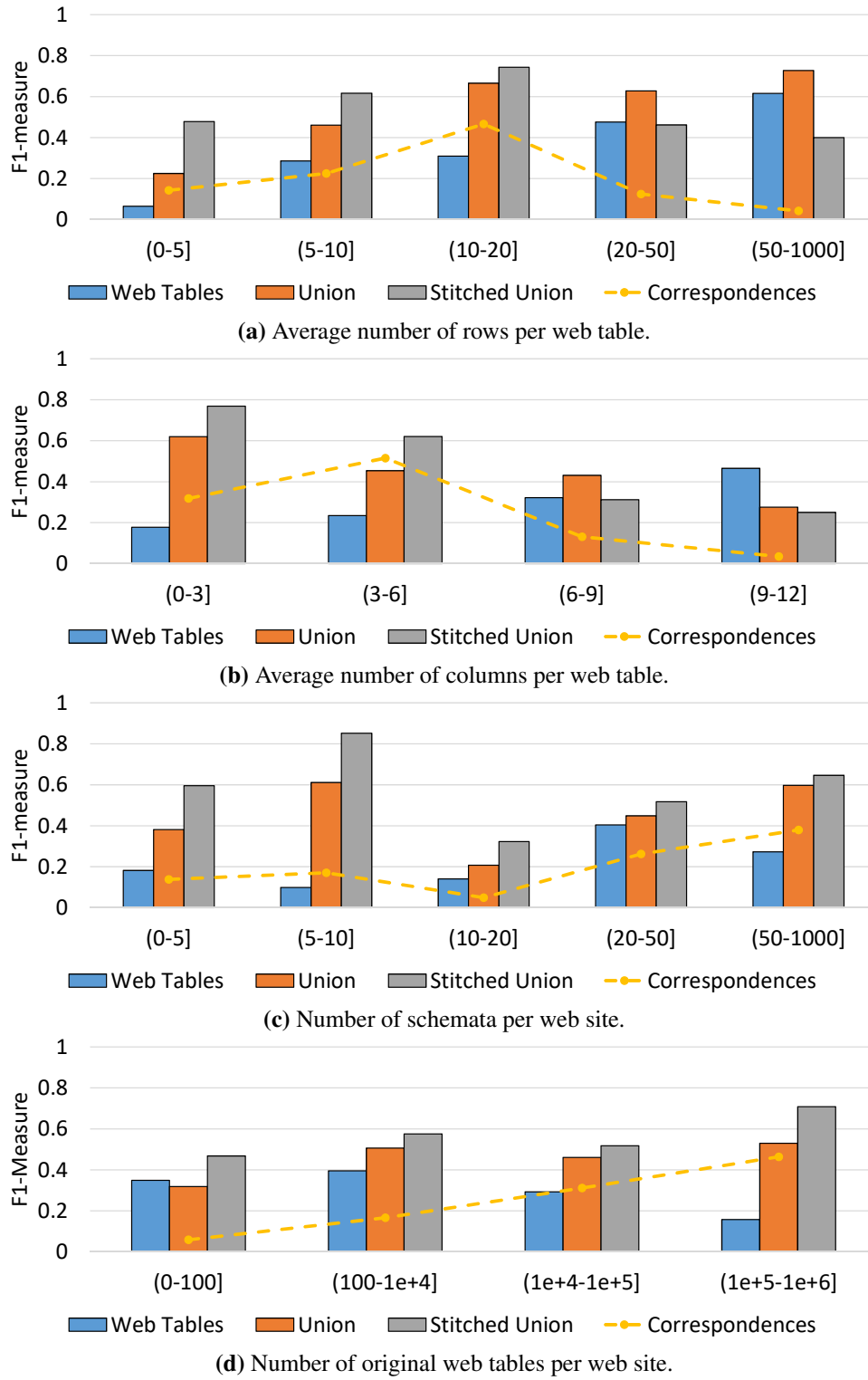
tables and the 75% mark is reached at 42 union tables per website (series “*Union Tables*”). Creating the stitched union tables for all web sites with up to 1 000 union tables further reduces the amount of tables to 104 221.⁴ More than 50% of the original tables are now provided by web sites with a maximum of only 3 stitched union tables and 75% of the original tables by websites with less than 7 stitched union tables (series “*Stitched Union Tables*”).

Table Characteristics

Another important aspect is how effective the stitching methods are for web sites that publish web tables with different characteristics. Hence, this section analyses for which table characteristics the stitching methods result in the largest benefits.

Figure 7.9 shows the semantic table interpretation schema matching performance with respect to four characteristics: average number of rows, average number of columns, total number of schemata, and total number of tables. These characteristics are determined from the original web tables for each web site, so this profile can be used to determine whether a stitching procedure should be applied or not. The bars in Figure 7.9 show the F1-measure for the random 1 000 table sample and the line indicates the fraction of all correspondences for each bin. Figure 7.9a shows that stitching is effective for small web tables with up to 20 rows, which is the main argument for applying a stitching procedure. For higher numbers of rows, the results of using stitched union tables are worse than the results of just using union tables. These are web tables which are already large enough for matching them directly, so every error that is introduced by the union table stitching procedure has a negative impact on the result. Figure 7.9b shows that a similar trend can be observed for the number of table columns. With up to 6 columns,

⁴A threshold of 1 000 union tables per web site is used to exclude web sites that cannot be handled by the method, such as Wikipedia. The resulting dataset contains 94.5% of all web tables in the original dataset.

**Figure 7.9:** Influence of table characteristics on matching performance.

the stitching approach is effective, but for tables with more columns the performance drops. Figure 7.9c shows that stitching improves the results regardless of the number schemata per web site. However, the advantage over only using the union tables is highest for up to 20 schemas. Figure 7.9d shows that stitching is an improvement regardless of the number of original web tables provided by a web site, but there is a trend that the improvement increases with the amount of web tables.

7.5 Conclusion

This chapter introduced a method for stitching web tables from the same web site based on their structural similarities and schema matching methods. First, related work on the merging of web tables and holistic schema matching methods was introduced in Section 7.2. Then, the distribution of re-used schemata in the Web Data Commons Web Tables Corpus 2015 was analysed and a holistic schema matcher that allows the handling of overlapping, but not exactly matching web tables was introduced and used to define two methods for web table stitching in Section 7.3. These methods were then experimentally evaluated and analysed with respect to their impact on improving the quality of semantic table interpretation methods in Section 7.4.

This chapter made the following contributions:

- **Re-Evaluation of Semantic Table Interpretation:** A re-evaluation of the performance of schema matching tools for semantic table interpretation on a random sample of web tables showed that the performance for small web tables is much worse than estimated by existing gold standards. The performance of T2K Match of only 0.24 in F1-measure for the schema matching task is not comparable to the score of 0.7 achieved on the T2D gold standard. An error analysis revealed that this is mainly due to very small web tables, which are strongly under-represented in such gold standards.
- **Data Profile of Schema Re-Use:** A data profile of the WTC 2015 that showed that the majority of the schemata of web tables are re-used frequently by other web tables on the same web site. This indicates that most web tables are created by templates in the back-end of the web servers and can be combined to form larger tables. The data profile showed that a 83% of all web tables have a schema that is reused by at least 100 other web tables on the web site.
- **Stitching Method:** Based on the profiling and evaluation results, a method was presented that stitches (combines) small web tables into larger tables based on their schema and holistic schema matching techniques. The application of this method reduced the size of the corpus from more than 5 million web tables to as few as 104 thousand stitched tables. An evaluation of existing matching algorithms on a random sample of web tables showed that

the quality of the results can be improved from 0.24 to 0.62 in F1-measure by applying the stitching method before running the matching algorithm. A detailed analysis of the improvements further showed that the improvements are largest for web tables which have up to 20 rows and up to 6 columns, i.e., the stitching method is most effective if the original web tables are small.

Both the distribution of schema re-use frequencies and the reduction in the absolute number of tables after applying the stitching method show that the majority of the data in the used web table corpus resides in web tables that are structurally similar and likely automatically generated. This finding changes the way we look at web tables and approach tasks related to extracting and integrating the data that they provide: Rather than treating every web table as an individual data source, they must be considered as views over a larger database in the back-end of web sites. Methods using web tables should hence use a model that recognises this generative process and try to reconstruct possible representations of this back-end database. Due to the structural similarity and template-driven generative process, this reconstructing can be as simple as creating the union of all web tables with the exact same schema.

The experiments with holistic schema matching further showed that the idea of a back-end database can also be used to construct constraints on the schema correspondences among web tables and hence improve the quality of schema matching approaches. Using the stitching method as pre-processing step before applying existing methods for semantic table interpretation resulted in significant gains in quality.

Finally, a manual evaluation of the triples that were extracted after applying the stitching method and T2K Match on the random sample of web tables showed an accuracy of 59%, which improves the quality of extracted triples by more than a factor of 2, compared to the 24% that were measured in Chapter 6. This indicates that web tables are less noisy than previously assumed. However, this requires holistic approaches and cannot be achieved by methods that process web tables individually without considering the source web site from which they were originally extracted.

Part III

Extending the Schema of a Knowledge Base

Chapter 8

Schema Extension

8.1 Introduction

The topical profiling of the Web Data Commons Web Tables Corpus 2012 in Chapter 6 revealed that the majority of the columns in web tables, where classes and entities can be recognised, cannot be matched to the properties in the DBpedia knowledge base. This indicates a large potential for extending the knowledge base’s schema, as the properties of the recognised entities that are stated in the columns of the web tables are not yet covered by the knowledge base. However, not all columns in the web tables represent an attribute that can be added to the knowledge base as a meaningful property.¹ Schema-extension methods hence try to estimate the relevance of the newly discovered attributes for existing classes using measures such as frequency, consistency, or coherency. The result is a ranked list of attributes, from which a human end user can select suitable new attributes that can be modelled as properties in the knowledge base. For the augmentation of a knowledge base, however, it is further necessary that triples can be generated to populate the new properties and that these triples can be understood without additional information from the original data source. So far, this second criterion is not specifically addressed by state-of-the-art methods that use web tables as data source.

In the related work, most studies of the schema-extension task focus on using unstructured text to discover the names of new attributes [Ravi and Paşca, 2008, Lee et al., 2013, Gupta et al., 2014]. In this case, an extracted attribute is considered relevant if it relates two entities or an entity and a literal value in a given sentence. Using web tables as data source, this judgement of relevance is a more complicated task, as there is no natural language sentence expressing the intended relationship and because the attributes that are represented by the columns in a web table can be

¹In this context, an “*attribute*” is an element in a relational schema and can be represented by columns in potentially many different web tables (see Chapter 2). The element corresponding to an attribute in a knowledge base is a “*property*”, which can be used in the predicate position of triples (see Chapter 3). For better readability, this chapter will use the term “*attribute*” for both, the elements of relational schemata and the properties in a knowledge base.

part of a relation of arbitrary arity. Existing studies of the schema-extension task using web tables mostly ignore this fact and focus on creating a ranking mechanism for extracted attributes that ranks frequently occurring attributes highly [Cafarella et al., 2009, Das Sarma et al., 2012, Gupta et al., 2014, Yakout et al., 2012]. The focus of these methods is on creating high quality lists of semantically related attributes, which can then be inspected and integrated by an end user. These methods, however, do not guarantee that meaningful triples for knowledge base augmentation can be created from the new attributes as additional, explanatory attributes might be missing.

This chapter analyses the potential of web tables for the schema-extension task with respect to two aspects: finding relevant attributes for extension and understanding the relationship between these attributes and the subject column of the respective web tables. First, several methods for finding relevant attributes are experimentally compared, showing that only using relevance as criterion is insufficient to guarantee that interpretable triples can be generated. Second, a categorisation scheme for attributes is introduced which allows for the judgement of attributes based on their relationship with the detected subject column of the web table. If this relationship is binary, the values of the attribute can be understood without additional information, and this categorisation hence enables the selection of attributes for which interpretable triples can be generated with the existing methods. The profiling of the WTC 2012 with respect to these categorisations reveals that the majority of the attributes which could not be matched to the knowledge base do, however, not contain columns that are in a binary relation with the subject column.

The contributions of this chapter are:

- **Attribute Ranking Methods:** Several state-of-the-art methods for attribute ranking are extended to consider correspondences between the web tables and a target knowledge base created by T2K Match and correspondences among the web tables, which are created by additional schema matching methods. These correspondences improve the estimation of co-occurrence frequencies, the main feature for attribute ranking, as attribute counts are not solely based on exact string matches of column headers. The experimental evaluation shows that the extended methods outperform the methods based on exact string matching.
- **Evaluation Methodology:** An experimental comparison of two evaluation methodologies for schema extension shows that the commonly used evaluation approaches based on relevance are problematic and can lead to inconclusive results. These results further indicate that the highly ranked attributes produced by the current methods are not guaranteed to be interpretable without additional information from the original web pages.
- **Column Categorisation:** The definition of a categorisation scheme as an indication for the completeness of discovered attributes and the design and evaluation of a classifier for this scheme are presented. Through the appli-

cation of this classifier, a data profile of the WTC 2012 according to this categorisation scheme is created, which shows that most columns are not in a binary relation with the detected subject column.

This chapter is organised as follows. Section 8.2 introduces related work for attribute ranking and schema extension. Then, Section 8.3 defines and experimentally compares several attribute ranking methods. Finally, Section 8.4 introduces a categorisation scheme for attributes, presents a classifier for this scheme, and profiles the Web Data Commons Web Tables Corpus 2012 according to these categories.

The work presented in this chapter, i.e., the comparison of schema-extension methods and the categorisation of attributes, has previously been published in [Lehmberg and Bizer, 2016, Lehmberg and Hassanzadeh, 2018].

8.2 Related Work

This section introduces the related work for schema extension. Generally, three different variants of the schema extension task can be identified: (1) find values for new attributes by keyword query (“*table extension*”), (2) find new attribute names (“*attribute discovery*”), and (3) find attribute synonyms (“*synonym discovery*”).

The first variant accepts a data table and a keyword query as input and tries to populate a new attribute that matches the query with values. This variant also solves the slot-filling task and is mostly chosen in combination with web tables as data source, which contain attribute values along with the attribute names. The second variant has been widely explored for unstructured text as data source and only focuses on the discovery of new attribute names. The third variant has the goal of finding new synonyms for known attribute names and often the second and third variant are approached jointly.

In this section, first schema discovery from general web sources is discussed. These are methods that use the content of web pages, search engine query logs, and other sources of data from the Web to find new attributes. Afterwards, specific methods for schema extension in web tables are introduced. In both cases, also the used evaluation methodology, which can differ considerably between the different approaches, is discussed.

Schema Extension from Web Sources

This section presents different types of approaches to extract attributes from web sources. These sources can be web pages, search engine query logs, or existing knowledge bases. All methods first extract candidate attributes from their sources and then apply a ranking mechanism based on the frequency of occurrences to determine relevant attributes. However, most of the approaches do not consider the extraction of values for the new attributes.

[Ravi and Paşca, 2008]. Ravi and Paşca propose to exploit the HTML structure of web pages for attribute extraction. Given a set of seed entities and attributes for a class, they obtain a small number of documents (50-200) by querying a search engine. From each document, candidate attributes are generated from “*emphasized HTML tags*”. The list of considered tags is not stated exhaustively, but it includes table columns “*td*” and table headers “*th*”. Based on the generated candidates, hierarchical extraction patterns (wrappers) are induced, which contain the names of the HTML tags of the candidate and all its parents. To find relevant attributes, two different ranking methods are proposed. The proposed ranking methods for the extracted attributes are *frequency-based* and *pattern-based* ranking. Frequency-based ranking scores attributes with the frequency of known entities for which the attribute can be extracted from the document collection. For pattern-based ranking, a pattern vector is created, which contains all extraction patterns that were induced for the candidate attribute, and the similarity of these vectors is used to rank the attributes. The method is evaluated using 40 classes with different numbers of instances and 5 seed attributes per class, with one human annotator assigning the labels “*vital*”, “*okay*”, or “*wrong*” to each extracted attribute. The results show that the pattern-based ranking performs better than frequency-based ranking and hand-crafted patterns. The authors further propose to extract values for the discovered attributes by selecting the content of the tag that immediately follows the attribute name as value. The value extraction is only evaluated for a small set of high-quality attributes, which are manually selected, and results in an accuracy of 74%.

[Lee et al., 2013]. Lee et al. extract attributes from Web documents, a search engine query log, and DBpedia and estimate their typicality for the respective classes. The extraction process is completely label-based and does not consider any attribute values. To merge attributes with the same semantic meaning, they use synonyms derived from Wikipedia. The authors define an attribute as typical for a class, if it is frequently mentioned as an attribute of the class and many of the instances of the class have this attribute. The typicality is estimated using frequency counts from the different sources from which the attributes were extracted. For the evaluation of their method, human annotators label 4846 attributes for 12 concepts with either “*very typical*”, “*typical*”, “*related*” or “*unrelated*”.

[Gupta et al., 2014]. The Biperpedia system extracts attributes from different sources such as Freebase, a search engine query stream, and Web documents. The attribute extraction is purely text based and attribute merging is done via synonyms and a misspelling correction. The attributes are classified into three categories: “*atomic-numeric*”, “*atomic-textual*”, and “*non-atomic*”. To evaluate the quality, the extracted attributes are ranked by their frequency in the query stream and in the text corpus and then shown to three human evaluators, who decide by majority voting whether the attribute is “*good*” or “*bad*”. Both ranking methods achieve a

high precision of 98% (by query) and 88% (by text) for the top 10 ranked attributes and 52% or 54%, respectively, for the top 5 000 ranked attributes. In a later work, Halevy et al. [Halevy et al., 2016] used the attributes extracted by the Biperpedia system to discover patterns that allow for a grouping of these attributes. They propose a grammar which is used to define rules with placeholders that enable an understanding of the attribute names. For example, the attribute name “*tyre price in Singapore*” can be grouped by the rule “*price*”, which identifies a component (“*tyre*”) and a market (“*Singapore*”). With their method, they are able to reduce the number of distinct attributes by a factor of 42 with a precision between 60% and 80%.

The approaches presented in this section use the document structure of web pages, textual patterns, and seeds given by a user or from a knowledge base to extract unknown attributes from web sources. The evaluation of the methods is performed by a rather subjective notion of relevance, which is especially shown by the different rating scales used in each approach. The focus of the methods is on finding relevant attributes and often no attribute values are extracted. This makes the results of these methods useful to understand user queries or other data sources on the Web, such as web tables. For direct knowledge base augmentation, however, the extraction of values and the integration of the new attributes with the knowledge base are necessary, but not addressed by the discussed methods.

Schema Extension from Web Tables

Approaches that use web tables instead of natural language sentences can access instance data of the discovered attributes easily, as it is contained in the same web table that contains the attribute names. Hence, approaches in this area often augment data source with new attributes and their values at the same time.

[Cafarella et al., 2008b]. The pioneering work using web tables to discover new attributes was presented by Cafarella et al. in 2008. They create an “*Attribute correlation statistics Database (AcsDb)*”, which contains attribute counts based on the column headers in a large corpus of web tables. From these counts, they estimate attribute occurrence probabilities. Applications for this database are a schema auto-complete function, synonym generation and a tool enabling easy join graph traversal for end users. The attribute discovery task is called “*schema auto-complete*” and ranks the attributes in the AcsDb based on point-wise mutual information [Church and Hanks, 1990] with the attributes that already exist in the schema. This is one of the approaches that are compared in the next section, where it is further extended by an additional matching step. The authors evaluate their method against ten test schemata that each contain all attributes that were mentioned by at least two out of six annotators who were presented the same task as the system. The system is given three tries to reproduce the test schema, and achieves an average recall of 46% in the first try and 62% with up to three tries.

[Cafarella et al., 2009]. The Octopus system, a system for data integration using web tables, defines an “*EXTEND*” operation. This operation requires the user to provide an input table and a keyword query. Based on this input, the system tries to find one or multiple tables that can extend the input table with attributes matching the keyword query. The system uses a search engine to find the tables and different instance-based matching approaches to determine which tables can be combined. A web table is considered “*joinable*” to a query table, if the Jaccard similarity between the join column in the query table and a column in the retrieved table exceeds a predefined threshold. In the case of multiple joined tables, tables are first clustered and the cluster with the largest coverage of entities in the query table is chosen. The clusters are created using the CENTER algorithm with a predefined similarity threshold. This is methodologically similar to the clustering step that is proposed for the matching of columns from different web tables in the next section. The extend operation is evaluated on 7 queries consisting of a table and an attribute keyword. On average, values for 33% of the rows in the query table are returned by the system.

[Das Sarma et al., 2012]. Sarma et al. use label- and instance-based schema matching methods to map web tables to a given query table. For their “*Schema Complement*” operation, they consider all unmapped columns and rank them using the AcsDb [Cafarella et al., 2008b] and the entity coverage of the input table provided by the user. Their goal is to rank web tables by their usefulness for the attribute discovery task. To rank the web tables, they consider co-occurrence statistics of attribute pairs from the query schema and the web tables, and measure the consistency of the schema that is created by adding an attribute to the query schema. This is another method that will be compared against other in the next section. The method is evaluated by eight annotators that assign a score to each web table that is returned for 18 different query tables. The results of the evaluation show only that the aggregation of attributes scores for each returned table is best with the “*sum*” function, but do not allow for a comparison to other methods as the rating scale is not explained.

The presented methods that use web tables for the discovery of new attributes usually take a user-provided table as input and either extend this table with new columns and their values or suggest additional attribute names based on the schema of the provided table. It is, however, unclear if these methods are also suitable for knowledge base augmentation, where all known entities of a class instead of a comparably small, user-provided query table should be augmented.

Evaluation Criteria

The related work that has been presented above is evaluated with respect to different tasks and criteria. Table 8.1 gives an overview over these tasks, the used criteria and their rating scales. The column “*Task*” indicates the specific sub-task

that was evaluated. Here, “*Attribute Discovery*” refers to the retrieval of attribute names given a class and entities of this class as input, “*Table Search*” refers to the retrieval of web tables given a keyword query or a table query, and “*Slot Filling*” refers to the retrieval of values for an attribute that is specified using a keyword query or generated through attribute discovery. The column “*Criterion*” indicates the quality criterion that was used to evaluate the proposed methods. The terms in this column are stated as in the original publication. Finally, the column “*Scale*” shows the range of possible values for the quality criterion as stated in the literature.

Table 8.1: Overview of tasks and evaluation criteria used in related work for schema extension from web tables.

Approach	Task	Criterion	Scale
[Ravi and Paşca, 2008]	Attribute Discovery	Correctness	Vital, Okay, Incorrect
[Ravi and Paşca, 2008]	Slot Filling	Correctness	Correct, Incorrect
[Lee et al., 2013]	Attribute Discovery	Relevance	Very Typical, Typical, Related, Unrelated
[Gupta et al., 2014]	Attribute Discovery	Precision	Good, Bad
[Cafarella et al., 2008b]	Table Search	Relevance	1 to 5
[Cafarella et al., 2009]	Table Search	Relevance	Yes/No
[Cafarella et al., 2009]	Slot Filling	Recall	0.0 to 1.0
[Das Sarma et al., 2012]	Table Search	Related	0 to 5
[Yakout et al., 2012]	Slot Filling	Precision, coverage	0.0 to 1.0

The comparison of criteria for evaluation shows that there is no common terminology that is used by all authors. Often, the used criterion is not further explained by the authors and assumed to be understood. Only Ravi and Paşca provide an explanation how to interpret their evaluation: “*An attribute is ‘vital’ if it must be present in an ideal list of attributes for the target class; ‘okay’ if it provides useful but non-essential information; and ‘wrong’ if it is incorrect.*”. Despite the different terminology, all approaches for attribute discovery and table search evaluate whether the retrieved attributes are relevant with respect to the target class or provided query table. A precise definition, however, does not exist and the specific decisions are often based on the consensus of several human annotators.

The relevance, or relevancy, of data is one possible dimension, among others such as accuracy, timeliness, accessibility or completeness, along which Wang and Strong propose to measure the quality of data [Wang and Strong, 1996]. In their conceptual framework of data quality, relevance is a dimension in the group of “*contextual data quality*”, which contains dimensions that are measured with respect to a task. Specifically, they define relevancy as “*the extent to which data are*

applicable and helpful for the task at hand". In the context of knowledge base augmentation, the task is to find new attributes for a target class, which is usually defined by a short natural language description. The relevance of new attributes is hence either determined based on human judgement, or through the evaluation of an application that uses the augmented knowledge base to solve a different task, as demonstrated by Gupta et al. [Gupta et al., 2014] who use the discovered attributes to annotate web tables.

Another dimension in the group of contextual data quality is "*completeness*", which is defined as "*the extent to which data are of sufficient breadth, depth, and scope for the task at hand*". As will be discussed in Section 8.4, this dimension is not sufficiently addressed by current methods, which can produce attributes and values that are lacking contextual data that are necessary to interpret the values.

8.3 Attribute Ranking

The first part of this chapter is concerned with attribute ranking. The purpose of attribute ranking methods is to order all candidate attributes for schema extension such that relevant attributes for a specific class or query table are ranked highly. Based on such a ranked list, users can then select appropriate attributes for their information needs.

Common approaches for attribute ranking consider the frequency and co-occurrence of candidate attributes with other, possibly known attributes. In addition to such frequency-based methods, this section also introduces methods for schema matching among web tables, which enables the estimation of frequencies for attributes which are represented by different surface forms of the same semantic intention, such as synonyms or different spellings.

The approach presented in this section consists of two steps. First, web tables are matched among each other and to the target knowledge base to identify columns that represent the same attribute and align them with classes and existing properties in the knowledge base. The second step ranks the attributes which are candidates for new properties based on a measure of relevance for augmenting the schema of the target knowledge base. The performance of this approach is studied empirically using web tables from Web Data Commons Web Tables Corpus 2012 for the augmentation of the DBpedia knowledge base.

8.3.1 Attribute Matching

This section introduces the schema matching approaches that are evaluated for the task of matching the web tables in a corpus among each other. In the following, several approaches of defining attribute similarity are defined, which will be used in the experimental evaluation.

Similarity of Known Attributes

For the attributes that correspond to existing properties in the knowledge base, a mapping from the web tables to the knowledge base is created. This mapping is created using T2K Match (see Chapter 5) as semantic table interpretation method. It defines which columns in the web tables correspond to which property in the knowledge base. By transitivity, all columns which correspond to the same property are considered to represent the same attribute.

Similarity of Unknown Attributes

Based on the mapping produced by T2K Match, all web tables can be grouped by their class annotation. Then, all un-matched columns in each group are matched among each other. For these columns, the following schema matching approaches are evaluated:

- **Label-based Matching:** The similarity of columns is determined by calculating the string similarity of their column headers. Variations of this approach are “*exact*”, which checks headers for equality, and “*string similarity*”, which applies the generalised Jaccard string similarity measure with Levenshtein similarity for token comparisons.
- **Instance-based Matching:** The similarity scores calculated by the instance-based schema matcher of the Helix System [Ellis et al., 2015] are used. This matcher compares the values of the columns of web tables using a set similarity measure. The configuration of this matcher using cosine similarity is referred to as “*Helix Cosine*”, and the configuration using containment similarity is referred to as “*Helix Containment*”.
- **Duplicate-based Matching:** The duplicate-based matching approach compares only those values of two columns, which are mapped to the same entity in the knowledge base. This means, two columns are similar only if they contain similar values for the same entities. These similarity scores are calculated by the duplicate-based matching component of T2K Match.

Similarity Graph Partitioning

After the calculation of the similarity scores, the matched columns must be partitioned into clusters that represent attributes. All columns which are mapped to an existing property in the knowledge base with a similarity score above a threshold are considered as the same attribute. However, for attributes which do not exist in the knowledge base, no such central property exists. The partitioning of columns that are not mapped by T2K Match is hence performed using different graph-based partitioning strategies [Hassanzadeh et al., 2009]. These strategies operate on the undirected graph that is defined by the columns as nodes and their similarity scores as weighted edges. The following partitioning strategies are evaluated:

- **Connected Components:** A connected component in a graph contains all nodes which are reachable via an edge or a path. This means that different components in a graph have no connections among any of their nodes. With this strategy, every connected component forms a cluster and all columns corresponding to the nodes in the cluster represent the same attribute.
- **Center:** The “*Center*” algorithm creates star shaped clusters by selecting one node as the centre and assigning all nodes that are connected by an edge to the same cluster. The clustered nodes are then removed from the graph and the algorithm starts over until all nodes are assigned to a cluster.
- **MergeCenter:** The “*MergeCenter*” algorithm is similar to the Center algorithm, but has one extension. This extension is that if a node is similar to the centres of two different clusters, these clusters are merged.

8.3.2 Ranking Strategies

After defining attribute matching, the different attribute ranking methods are introduced in the following. All compared ranking methods are based on attribute occurrence probabilities, which are defined in the following according to Cafarella et al. [Cafarella et al., 2008b].

Let a schema $s \in S$ be a set of attributes and S be the set of all schemata. A table has this schema if its columns correspond to the attributes based on the schema mapping, regardless of their order and column header. Let $\text{freq}(s)$ be the number of tables with schema s in the corpus and $\text{attr_freq}(a)$ be the number of tables that contain attribute a :

$$\text{attr_freq}(a) = \sum_{s \in S \wedge a \in s} \text{freq}(s) \quad (8.1)$$

Then, the probability of encountering a in any table in the corpus is:

$$p(a) = \frac{\text{attr_freq}(a)}{\sum_{s \in S} \text{freq}(s)} \quad (8.2)$$

The number of tables that contain two attributes a_1, a_2 is defined analogously as $\text{attr_freq}(a_1, a_2)$. The conditional probability of seeing attribute a_1 given a_2 is:

$$p(a_1|a_2) = \frac{\text{attr_freq}(a_1, a_2)}{\text{attr_freq}(a_2)} \quad (8.3)$$

And the joint probability is:

$$p(a_1, a_2) = \frac{\text{attr_freq}(a_1, a_2)}{\sum_{s \in S} \text{freq}(s)} \quad (8.4)$$

Using these attribute occurrence probabilities, the ranking methods can now be defined. A higher score indicates a higher relevance of the attribute for the schema-extension task.

Conditional Probability based on Class. Measures the probability of seeing the attribute a given that the web table is mapped to the class c in the knowledge base [Lee et al., 2013]. If $\text{attr_freq}(a, c)$ is the number of tables mapped to c that contain a , the conditional probability of encountering an attribute based on the class can be defined as in Equation 8.5, where S_c is the schema of class c .

$$p(a|c) = \frac{\text{attr_freq}(a, c)}{\sum_{a_2 \in S_c} \text{attr_freq}(a_2, c)} \quad (8.5)$$

Schema Consistency. This measure reflects the likelihood of seeing a new attribute together with the existing attributes of a knowledge base class [Das Sarma et al., 2012]. This measure considers all known attributes which co-occur with the new attribute a , i.e., the more known attributes co-occur, the higher the score.

$$\text{SchemaConsistency}(a, s) = \frac{1}{|s|} \cdot \sum_{a_2 \in s} p(a|a_2) \quad (8.6)$$

Schema Coherency. Based on Point-wise Mutual Information (PMI), schema coherency is the average of the PMI scores of all possible attribute combinations between a new and all existing attributes of the class that should be extended [Cafarella et al., 2008b]. The PMI score of two attributes is positive if the occurrences of the attributes are positively correlated, zero if they are independent, and negative if they are negatively correlated.

$$\text{SchemaCoherency}(a, s) = \frac{1}{|s|} \cdot \sum_{a_1 \in s} \text{pmi}(a_1, a) \quad (8.7)$$

$$\text{pmi}(a_1, a_2) = -\frac{1}{\log p(a_1, a_2)} \cdot \log \frac{p(a_1, a_2)}{p(a_1) \cdot p(a_2)} \quad (8.8)$$

8.3.3 Evaluation

This section presents an evaluation of the introduced schema matching and attribute ranking methods. First, the schema matching methods are evaluated on the T2D gold standard (see Section 5.3). Second, the results of the introduced attribute ranking methods are evaluated on web tables from the T2D gold standard and then compared to the results obtained on the web tables in the Web Data Commons Web Tables Corpus 2012.

Schema Matching

This section shows the evaluation of the different schema matching and partitioning approaches introduced in Section 8.3.1. The goal of the evaluated approaches is to create clusters of web table columns which are semantically equal.

For this evaluation, the T2D gold standard is used in a modified version. The original gold standard contains mappings from the web table columns to properties in the knowledge base. To create column clusters, all web table columns which are mapped to the same property are merged into one cluster. The results of the schema matching methods for unknown attributes and the partitioning approaches are then compared to these reference clusters.

Figure 8.1 shows the performance of all matching and partitioning approaches for different similarity thresholds. While all matching approaches achieve comparable precision, only the label-based matcher achieves a high recall. The low recall of the instance-based and duplicate-based matchers can be attributed to the small size of the web tables. These matchers compare the overlap of values between columns of two tables, and hence cannot find a match if there is low or even no overlap in the values. For example, two web tables that list countries and their capital cities do not have any overlap, if one web table lists countries with names starting from A to M and the other one from N to Z.

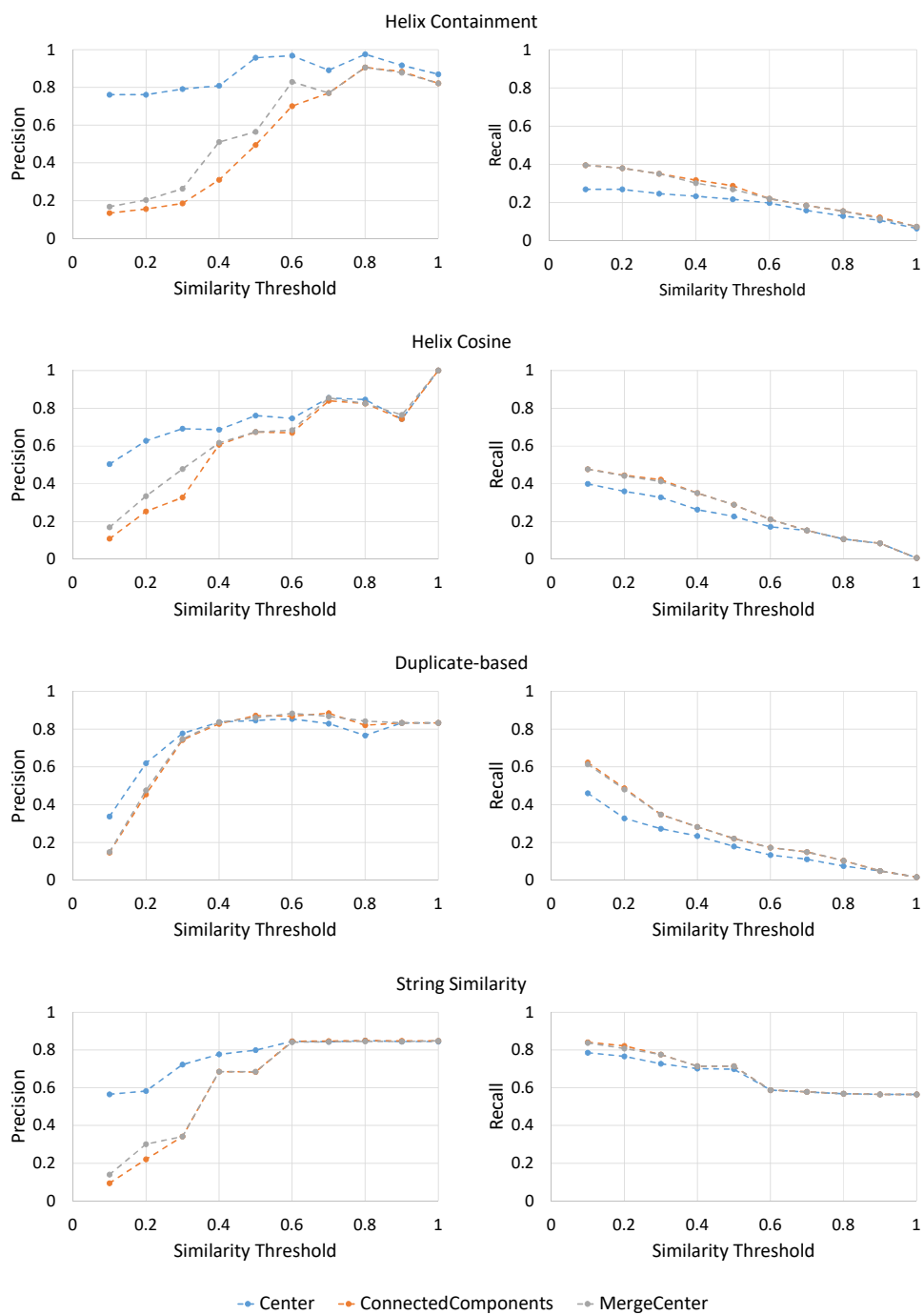
The high recall of the label-based similarity can indicate that the web tables in the used gold standard are very homogeneous. The T2D gold standard was designed for the evaluation of semantic table interpretation tasks, and the homogeneity or heterogeneity of columns in different web tables was not a design criterion. Combining the instance-based and label-based approach into a hybrid matcher did not significantly improve the performance compared to the label-based approach, which indicates that the matches that are found with instance-based similarity can also be found by label-based similarity. Concerning the partitioning approaches, Center achieves the highest precision and has a slightly lower, but comparable recall to the other approaches.

Attribute Ranking on the T2D Gold Standard

The first set of experiments for attribute ranking is performed on the T2D Gold Standard (see Section 5.3), which was originally developed to evaluate systems for semantic table interpretation using DBpedia as target knowledge base.

Evaluation of Ranked Lists. The evaluation of the attribute ranking methods follows the methodology used in the related work. For a subset of the T2D tables, those mapped to the `country` class, all columns are manually annotated with either “*relevant*” or “*not relevant*” with respect to the target class. A relevant attribute is an attribute that is recognised by the annotator as an attribute of the entities of the target class, such as “*population*” for countries. Attributes which are not relevant are such which are not by themselves an attribute of these entities, for example “*date of information*”. In total, the annotated subset contains 207 columns, of which 86 are annotated as “*relevant*”. With these annotations, the performance of the different ranking methods is evaluated.

Figure 8.2 shows the precision@k and recall@k achieved by the different ranking approaches. In addition to the ranking methods described in Section 8.3.2, a

**Figure 8.1:** T2D: Evaluation of matching methods for unknown attributes.

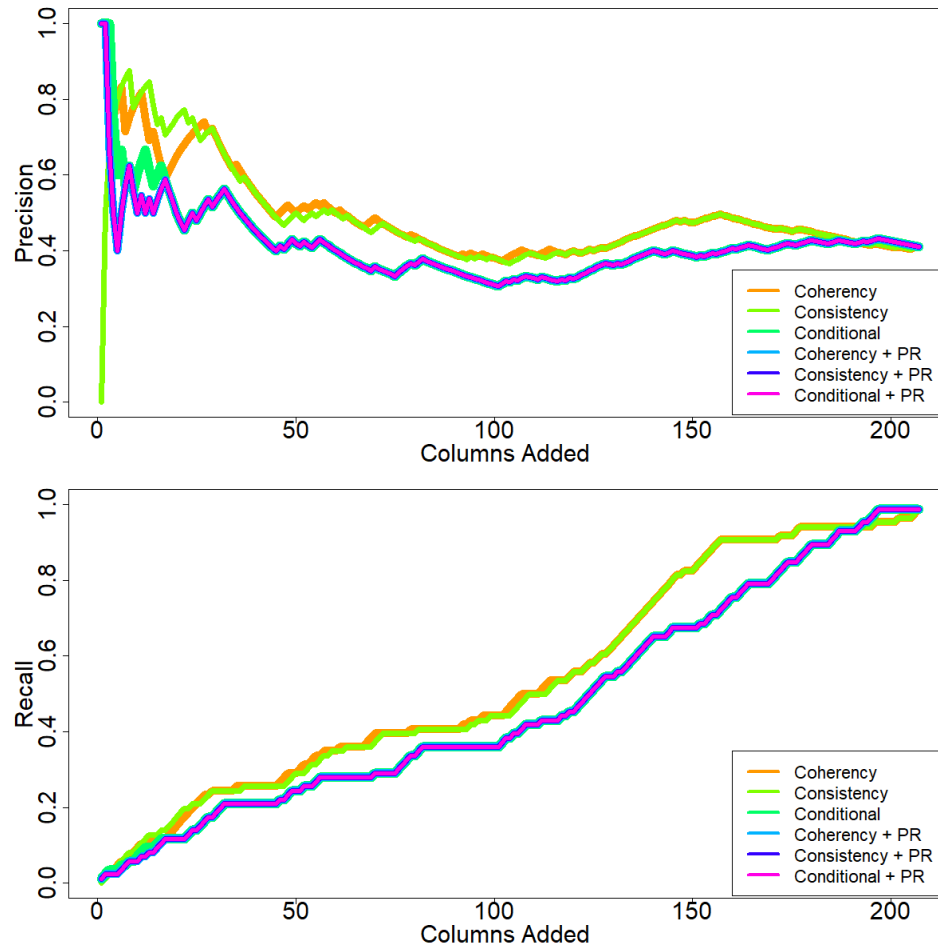


Figure 8.2: T2D: Precision@k and recall@k achieved by the different ranking methods using the duplicate-based matcher.

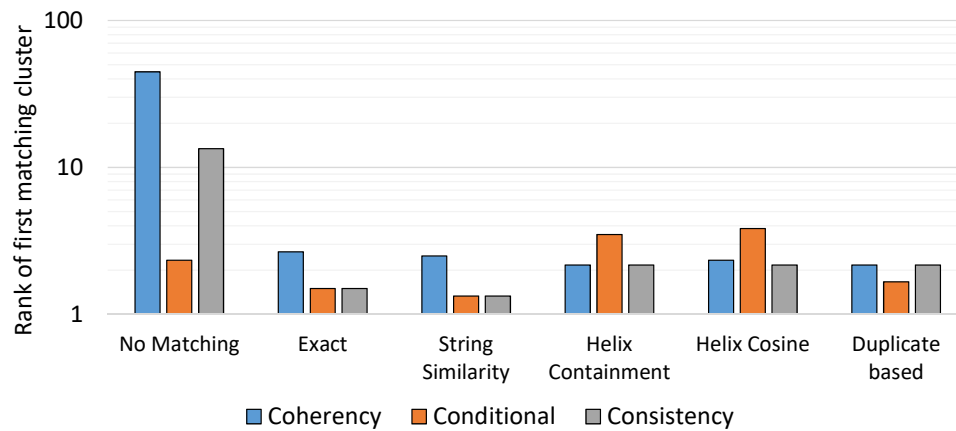


Figure 8.3: T2D: Rank of the first cluster matching the removed attribute.

variant of each of the ranking methods that is weighted by PageRank is evaluated. The intuition is that web pages with a high PageRank likely contain useful content and hence the web tables on these pages might also contain relevant attributes. The used PageRank values are obtained from the publicly available Common Crawl WWW Ranking.² For each cluster of columns, the maximum PageRank of all source web pages is multiplied with the score that was calculated by the ranking method.

Among the different ranking methods, schema consistency performs best, followed by schema coherency. The variations with PageRank perform worst, which might be caused by the rather small number of web sites in the gold standard. The results show that the ranking methods schema consistency and schema coherency produce many useful attributes until a list size of 25, but afterwards the precision drops and converges around 0.5. This is most likely caused by the low frequencies of many attributes in the dataset, which indicates that these ranking methods cannot be used for infrequent attributes.

Evaluation of Individual Attributes. The assessment of the relevance of an attribute can be subjective. Hence, in a different experimental design, one existing attribute from the knowledge base is removed for several classes. As this attribute was already existing, it can objectively be said that it is relevant. The evaluation then measures the position of the first cluster that represents this attribute according to the different ranking methods. The following classes and attributes are used in this experiment: Company (industry), Country (population), Film (year), Mountain (height), Plant (family), VideoGame (genre).

Figure 8.3 shows the average rank of the first cluster which matches the removed attribute for all ranking methods and matchers. The group “No Matching” shows the result of neither using correspondences to the knowledge base nor any

²<http://wwwranking.webdatacommons.org/>

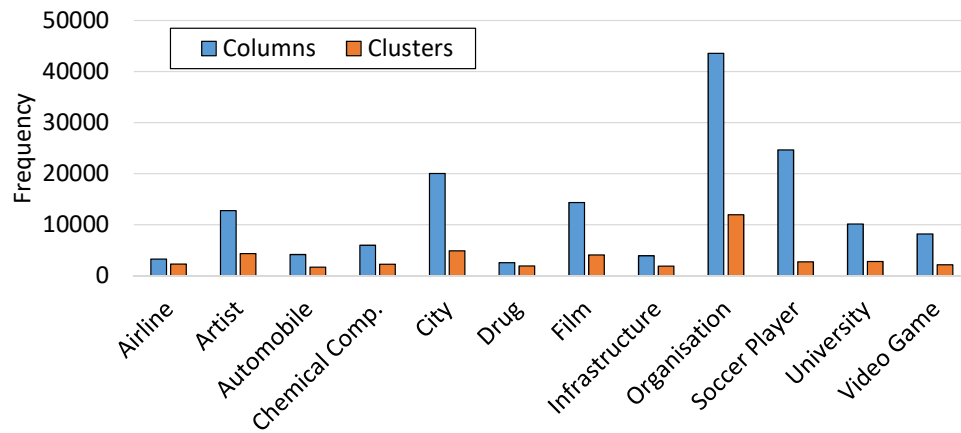


Figure 8.4: WDC 2012: Number of candidate columns and clusters for schema extension.

of the matching approaches, i.e., attributes are equal only if their column headers match exactly. The bad results for this setting without prior mapping knowledge show the importance of schema matching before calculating the ranking functions: for the coherency and consistency measure, the results are much worse than in any other configuration. Without mapping knowledge, attribute frequencies are underestimated, and the respective attribute is ranked too low.

The best configurations are exact string matching and string similarity matching with the conditional and consistency ranking methods. The worse performance for the instance-based matchers is likely due to their low recall, as seen in Figure 8.1. Comparing the performance of the ranking methods across the different matching approaches shows that the schema consistency measure is often the best choice, confirming the results of the previous experiment. This does, however, not hold for the schema coherency measure, which often produces worse results in this experiment. The results for schema coherency are hence inconclusive.

Attribute Ranking on the WTC 2012

The experiments described above are now repeated on all web tables in the Web Data Commons Web Tables Corpus 2012 which were matched to DBpedia, as described in Chapter 6. The instance-based matching approach based on the Helix system is, however, excluded as it performed worst on the T2D gold standard. To give an overall impression of the corpus, Figure 8.4 shows the number of candidate columns and clusters for schema extension for selected classes. These numbers show the large amount of potentially new attributes that can be found in the corpus.

Evaluation of Ranked Lists. As there is no gold standard for the full corpus, the top 15 ranked clusters for each ranking method for several classes are manually annotated with either “*relevant*” or “*not relevant*”. Figure 8.5 shows the performance

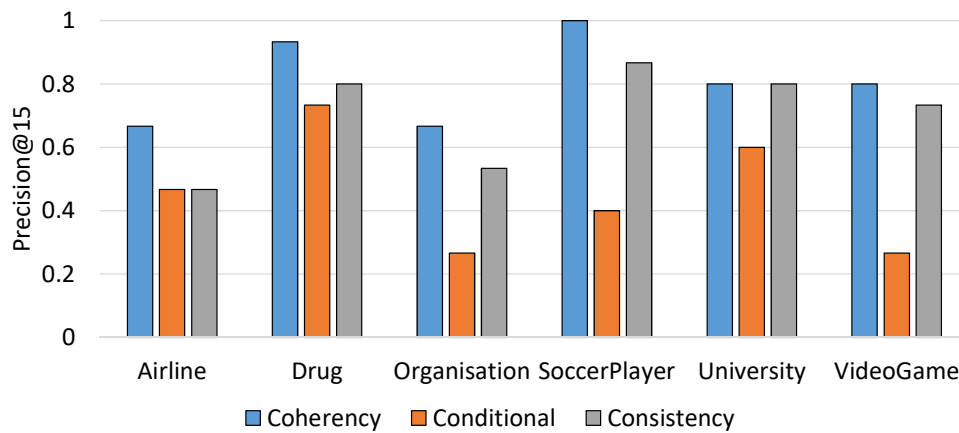


Figure 8.5: WDC 2012: Precision@15 for the attribute ranking experiment.

of each ranking method for all classes in terms of precision@15. The results show again that the schema coherency and consistency measures outperform the conditional measure. This indicates that attribute co-occurrence is a stronger signal than pure frequency of attributes, even if conditioned with a class from the knowledge base. In contrast to the results of this experiment on the T2D gold standard, schema coherency now outperforms the schema consistency measure.

Evaluation of Individual Attributes. Again, to have a more objective view on the results, one attribute is removed from DBpedia as before and the position of the top-ranked attribute cluster which matches the removed attribute is compared. Figure 8.6 shows the rank of these clusters by matcher and by ranking method.

Concerning the matching approach, it can be seen that the string similarity and duplicate-based matchers achieve comparable results. The difference to the same experiment on the gold standard is that here a much larger number of tables is taken into account and hence more variety and a more realistic sample of the data quality is used for evaluation. For the duplicate-based matcher this can mean that more tables actually contain duplicates, resulting in more discovered matches. Comparing both of the matching approaches to a baseline approach (“*No Matching*”), which does not use the prior knowledge of the mappings to the knowledge base, it can again be seen that the ranking results are worse, except for the schema coherency method.

Concerning the ranking methods, the conditional ranking performs best. In contrast to the experiment on the T2D gold standard, the consistency ranking performs worse and is no longer the best choice. It is, however, consistent that the schema coherency measure performs worse in this evaluation setting than in the setting where the all top ranked attributes are evaluated. This indicates that the attributes which were removed from the knowledge base might introduce a bias and have different characteristics than the attributes which are ranked highly by the coherency approach. This is supported by the performance of the conditional

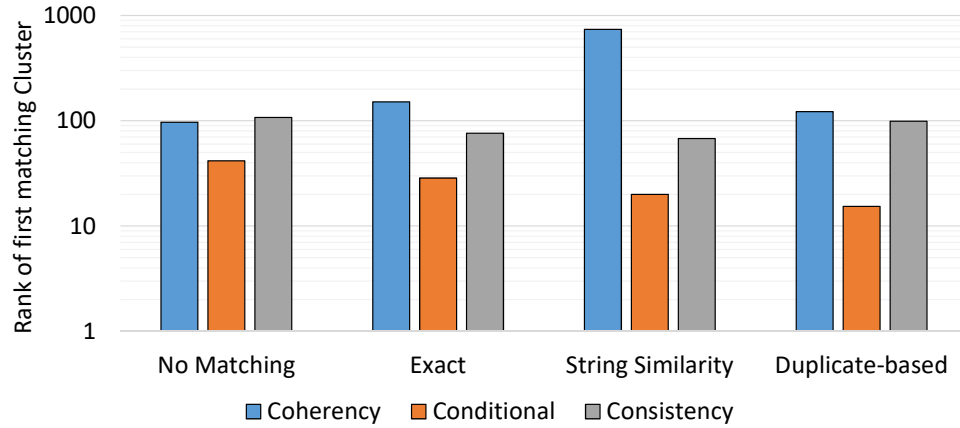


Figure 8.6: WDC 2012: Rank of the first cluster matching the removed attribute.

ranking method, which performs best for the removed attributes, but worst for the evaluation of the top-15 ranked attributes. A possible reason is that the removed attributes are among the most common ones for the respective classes, and thus occur most frequently in the web tables on the large corpus, which would explain the good performance of the conditional ranking method.

8.3.4 Result Analysis

The experiments presented in the previous sections have shown how the different ranking methods proposed in the literature perform on two different datasets, the T2D gold standard and the Web Data Commons Web Tables Corpus 2012. Further, it was demonstrated that schema matching among the attributes improves the results in most cases.

The evaluation of the attribute ranking methods was conducted in two different settings. First, a set of columns was manually labelled as either “*relevant*” or “*not relevant*”. While this or a similar approach is used in most of the related work, it is subjective and a precise definition of relevance is missing. Second, an objective evaluation was performed by considering several existing attributes in the knowledge base as non-existing. While this approach overcomes the problem of a missing definition for relevance, it introduces a bias: The attributes which already exist in the knowledge base can be considered to be the most relevant attributes for the respective classes. Hence, it can be expected that many web tables contain these attributes and as a result, the evaluation is limited to very frequent attributes. This could explain the outstanding performance of the purely frequency-based conditional ranking strategy.

For the other highly ranked attributes in this second evaluation setting, it is unclear whether they are suitable for schema extension or not. Hence, a qualitative analysis is performed based on the results produced by the different methods on the WTC 2012. Tables 8.2 and 8.3 show the highest ranked attributes for the classes

Drug and Airline, respectively. They show examples which are obviously relevant and similar to the properties which already exist in the knowledge base, such as “*common side effects*” and “*cas no*” for Drug or “*website*” and “*country*” for Airline.

For other attributes, a lack of context prevents the extraction of interpretable triples for a knowledge base. For example, the “*price*” of a Drug is not helpful for any task without knowing the brand or packaging size. Also, the “*terminal*” of an Airline cannot be used to satisfy a user’s information need if the corresponding airport is not known. But such missing information cannot be added through an improved ranking method. It is rather a signal that only focusing on relevance as quality criterion is insufficient.

For some of these attributes, the additional data that is missing is contained in other highly ranked attributes, such as “*brand name*” for drugs, which means that there exist cases in which more than one attribute needs to be added at the same time. This shows that data from web tables that is selected only according to relevance can be incomplete. In a knowledge base augmentation scenario, such attributes, which are missing explanatory context, cannot be transformed into interpretable triples, and are hence not suitable.

Table 8.2: Highest ranked attributes for class Drug.

Conditional	Consistency	Coherency	Low Rank
brand name	pharm. company	pharm. company	orders
quantity	type	how supplied	partial
price	synonyms	class	partner
consult	chemical type	description	r4
ship	half life	half life	r5
comments	main brand name	main brand name	r6
class	molecular formula	off-label uses	<enc. error>
common side effects	off-label uses	saliva	<enc. error>
common brand-name	saliva	washout time	<enc. error>
products			
dose	urine	synonym	<enc. error>
symptoms treated	washout time	dea number	<enc. error>
cas no	skeletal formula	therap cat	<enc. error>
trade name	other name	chemical type	<enc. error>
brand	catalog number	skeletal formula	<enc. error>
type	size	therapeutic use	<enc. error>

Table 8.3: Highest ranked attributes for class *Airline*.

Conditional	Consistency	Coherency	Low Rank
#	#	awb prefix	wwwaircorsicacom
website	end	end	year-to-date
terminal	awb prefix	iata prefix	yr1
telephone	country	start	yr2
rank	iata prefix	telephone	yr3
phone number	start	country	yr4
%	telephone	base airport	yr5
country	fleet	terminal	zeit
phone	base airport	acmi	zeme
end	%	flight no	exceptions
address	terminal	status	exemptions
1st bag	phone no	lounges	free items
2nd bag	air france	land	best low-cost airlines: worldwide
awb prefix	latin america	website	echanger des miles de destination
start	status	fleet	w

8.4 Attribute Categorisation

The result analysis in the previous section has indicated that the discovered attributes might depend on additional context information. This indicates that the quality criterion of relevance is not sufficient to find desirable attributes for schema extension and that completeness is an additional dimension that needs to be considered. The knowledge base augmentation task of schema extension can only be solved if the information that is contained in the new attributes is complete, i.e., can be interpreted without additional information from the original source. This is a hint that the used data model in semantic table interpretation methods is oversimplified. The frequently used model assumes that all columns in a web table are in a binary relation with the subject column, which means that no additional information is necessary to understand them. The fact that these assumptions lead to attribute candidates that show a lack of completeness indicates that there might be relations of higher arity in the web tables.

This section proposes a categorisation scheme for web table columns that allows to differentiate between columns which comply with assumption made by the current data model, i.e., represent binary relations with the subject column, and columns which violate these assumption and require additional information. Using this categorisation scheme, a manual profile based on a random sample of web tables is created, which indicates that n-ary relations, i.e., those that violate the current assumptions, make up the majority of columns. This manually annotated

sample is then used to train a classification model, which is applied to the Web Data Commons Web Tables Corpus 2012 to create a large-scale profile of column categories that confirms the initial results.

In a knowledge base augmentation use case, a possible application of this column categorisation classifier is to only consider columns which represent binary relations as candidates for new attributes. This reduced set of candidate attributes can then be ranked with the methods shown in the previous section, which results in relevant and complete new attributes.³

8.4.1 Categorisation Scheme

This section presents a detailed analysis of different categories for columns that occur in web tables. This categorisation focuses on the relation between a column and the subject column of the web table.

The data model for web tables that has been considered so far (see Chapter 5) and that is used by most of the related work states that every web table has a subject column which is a key of the web table and hence every other column represents a binary relation with the subject column. This model, however, is a simplification and does not acknowledge that a key in a web table might span multiple columns, which means that the table contains n -ary relations ($n > 2$), or that a column might not be in a relation with the subject column. An example for an n -ary relation is a web table with air fares: three columns “*from*”, “*to*”, and “*airline*” form the key and in combination with the attribute “*price*” a quaternary relation $\{from, to, airline, price\}$. An example for a column that is not in a relation with the subject column is a column that is only used as a layout element, which can be an empty column or one that contains the same content in every row.

The proposed categorisation distinguishes between columns which (1) are in a binary relation with the subject column and hence correspond to the data model, (2) are in an n -ary relation ($n > 2$) with the subject column and hence violate the assumptions of the data model, and (3) are not in relation with the subject column and are hence irrelevant for knowledge base augmentation. Figure 8.7 shows an overview of the proposed categories. The categories describe the columns and their relation with the detected subject column, i.e., a column can have a different category if a different subject column is chosen.

Binary Relation. A binary relation is a relation between two columns. The category *binary relation* is assigned to all columns which are in a binary relation with the subject column. This category is further divided into *time-independent* and *time-varying*. A binary relation is considered time-independent if its values do not change over time, for example a book’s author. Time-varying relations change their values over time, for example a country’s population. The decision to

³Note that the categories and the classifier are defined in terms of columns, not attributes, as the semantically same attribute can be represented by several columns which can be in different relations, i.e., columns representing the same attribute can be classified differently.

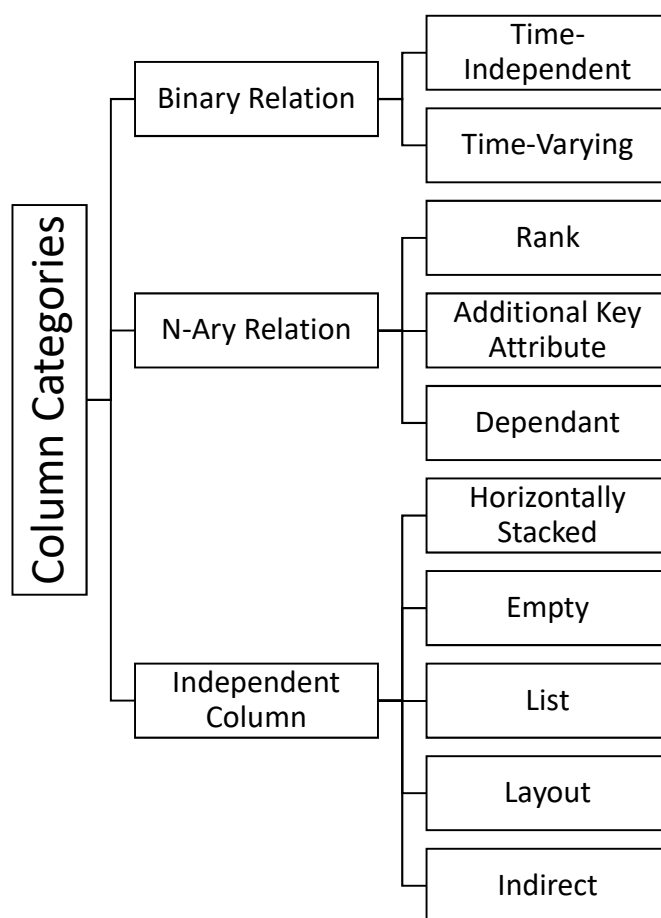


Figure 8.7: Category Hierarchy

treat time-varying attributes as binary relations is justified by the fact that existing approaches, which only consider binary relations, already consider timestamps as additional column metadata [Zhang and Chakrabarti, 2013, Oulabi et al., 2016] and the fact that a relation in a web table can be time-varying, even if the web table does not contain a column that states a value for the time dimension.

N-Ary Relation ($n > 2$). An n -ary relation is a relation among n columns. The category *n-ary relation* is assigned to all columns which are in an n -ary relation with the subject column and at least one additional key attribute ($n > 2$). The values of such a column can only be interpreted if all additional key attributes are known. Additional key attributes can be other columns or even be stated outside of the table (for example in the paragraph or heading before the table). Subcategories can be *rank*, *additional key attribute* and *dependant*. A *rank* is a numeric column that specifies an order of the rows in the table. It is an n -ary relation because at least one additional key attribute is required that specifies the ranking, for example the name

Men's Long Jump

Pos	Athlete	Nat	Mark
1	Phillips Dwight	USA	8.31
2	Al-Sabee Hussein Taher	KSA	8.30
3	Gaisah Ignisious	GHA	8.26
N-R	Sub	B-I	N-D

(a)

Airline	IATA	ICAO	Airline	IATA	ICAO
Adria Airways	JP	ADR	Hemus Air SP Ltd.	DU	HMS
Aegean Airlines	A3	AEE	Hola Airlines	BX	HOA
Air Berlin	AB	BER	InterSky	3L	ISK
Sub	B-I	B-I	I-HS	I-HS	I-HS

(b)

From	To	By Airlines	USD \$ Fare	Special Price
Kathmandu	Lukla	Yeti Airlines	113	Request Now
Kathmandu	Phaplu	Yeti airlines	107	Request Now
N-A	Sub	N-A	N-D	I-L

(c)

Countries

Afghanistan	Albania	Algeria	Andorra
Angola	Anguilla	Antarctica	Argentina
Armenia	Aruba	Australia	Austria
Sub	I-Li	I-Li	I-Li

(d)

Subject Column	N-ary Relation	Independent
	• Rank	• Horizontally
Binary Relation	• Additional Key	Stacked
• Time	Attribute	• List
Independent	• Dependant	• Layout

Figure 8.8: Example tables with column category annotation.

of a sports event in which athletes are ranked. An *additional key attribute* is part of an n-ary relation and specifies how to interpret the dependent column's values. An example can be seen in Figure 8.8c, where “From” and “By Airlines” are additional key attributes which together with the subject column “To” determine the value of “USD \$ Fare”. The subcategory *dependant* describes all columns in n-ary relations which are neither a rank nor an additional key attribute, i.e., which are the columns that depend on the subject column and the additional key attributes. Other than for binary relations, no dedicated distinction between time varying and time independent is made, as a time dimension is a special case of the more general *additional key attribute* category.

Independent Column. The category *independent column* is assigned to all columns which are not in a direct relation with the subject column. The first subcategory is *empty*, which describes columns that do not contain any data. The subcategory *indirect* is assigned to all columns that are transitively dependent on the subject column. The last three subcategories describe different types of noise that is specific to web tables. A *horizontally stacked* column is a column that is repeated in the same web table with the purpose to stretch the table horizontally instead of adding more rows. The result is that one row represents multiple tuples, which have no relation other than being placed together for visual reasons. An example is shown in Figure 8.8b. The same idea applies to *list* columns, which appear in tables that only contain the subject column multiple times. The sole purpose of such tables is to list a set of entities, hence they do not contain any relations (see Figure 8.8d). The last subcategory is *layout*, which describes columns that are inserted into tables for formatting reasons or contain navigational content (see the last column in Figure 8.8c, which only provides a link to another page on the same web site but does not contain actual data).

8.4.2 Manual Category Profiling

This section describes the creation of a category profile by manual annotation of a random sample of web tables from the Web Data Commons Web Tables Corpus 2012. This sample shows the frequency of each of the proposed categories and is used to train a classification model for the subsequent large-scale profiling of categories in the same corpus.

Sampling Method. The category profile is created from a random sample of 400 web tables (about 1 400 columns) from the WDC Web Tables Corpus 2012, which have been mapped to DBpedia as described in Chapter 6. All columns in these web tables are manually annotated with the categories described in the previous section. From this sample, 13 tables were excluded because they were either matched incorrectly or the annotator could not understand the content due to foreign languages.

The sample is constrained by the following conditions to ensure that the tables are useful for the schema-extension task, where evidence from multiple sources is required for reliable results (for example for the ranking methods presented in Section 8.3). Each web table must have a schema which appears on at least two different web sites. The schema of a web table in this context is the ordered set of column headers and the column data types. Further, the union of all tables with the same schema must contain at least 50 distinct entities and at least three of these entities must appear in tables from different web sites. These criteria have been chosen empirically to ensure a sufficient overlap in the data while not excluding too many of the tables in the corpus.

Table 8.4: Distribution of column categories in the manually annotated sample.

Category	% of all Columns
Binary	41.72%
Time-Independent	31.29%
Time-Varying	10.43%
N-ary	45.81%
Dependant	34.39%
Rank	8.74%
Additional Key Attribute	2.68%
Independent	12.47%
Horizontally Stacked	5.00%
Empty	4.23%
List	1.90%
Indirect	0.78%
Layout	0.56%

Category Profile. Table 8.4 shows the distribution of the column categories in the sample. Binary relations account for 42% of the sample, n-ary relations for 46%, and independent columns for 12%. The majority of the columns which are categorised as binary relation are time-independent (31% of all columns). For the n-ary relation category, additional key attributes appear very infrequently, with less than 3% of all columns assigned to this category. This implies that the additional key attributes are often located outside of the web table, which complicates their detection. For independent columns, the most frequent subcategories are horizontally stacked columns (5% of all columns) and empty columns (4% of all columns). This data profile shows that the majority of the columns in the analysed web tables do not comply with the assumptions of the used data model, as they are either in n-ary relations (46%) or not related to the subject column (12%).

8.4.3 Classification Model

The statistics of the presented data profile are a first indication that n-ary relations occur frequently in web tables and that the current data model is insufficient. This section describes how the manually annotated sample is used to train a classification model for the column categories which can then be used to create a large-scale profile and verify these findings. Due to the low frequency of several of the sub-categories in the sample, this model is restricted to the top-level categories binary, n-ary, and independent.

The remainder of this section first introduces the methodological foundation for the detection of the categories. Then, the specific features used in the classification model are defined. Finally, the evaluation of this classification model is presented.

Functional Dependencies in Web Tables

This section describes the methodological foundation for the features that are used in the proposed classification model. Specifically, different types of functional dependencies and their detection in the mostly very small web tables are described.

In sufficiently large web tables, the categories can be distinguished by discovering the functional dependencies (FDs) that hold on the web table. All columns A_b which are determined by the subject column A_e , i.e., $\{A_e\} \rightarrow \{A_b\}$, are in a binary relation with A_e . All columns $A_n \neq A_e$ which depend on or determine additional attributes, i.e., $X \rightarrow Y$ with $A_n \in X \cup Y$ and $A_e \in X$, are in an n-ary relation with A_e . All columns A_i which are not dependent on the subject column, i.e., $X \rightarrow \{A_i\}$ with $A_e \notin X$, are independent columns.

The problem with this approach is that most web tables are very small, and the discovered functional dependencies will in most cases confirm the assumption that all columns are in a binary relation with the subject column. The reason is that data-driven functional dependency discovery algorithms can only find n-ary relations if two rows with the same value for the subject column disagree on the value of another column. For example, in a web table with the ternary relation $\{country, date, population\}$, the FD $\{country, date\} \rightarrow \{population\}$ can only be discovered if the same country appears in two rows with different population numbers.

To avoid this problem, the data from multiple web tables is considered to discover the functional dependencies. To find matching web tables, the correspondences created with T2K Match (see Chapter 6) are used for all columns which can be mapped to DBpedia, and a label-based matcher is used for all other columns. Columns which cannot be matched to DBpedia are considered a match if their web table is mapped to the same class in the knowledge base and their column header and data type match exactly.

Using the set of matching columns, larger tables are constructed as follows: For each column A , the union of the subject column and column A from all web tables W_A containing A is created, as shown in Equation 8.9.

$$u_A = \bigcup_{t \in W_A} t[A_e, A] \quad (8.9)$$

Then, it is checked if the FD $\{A_e\} \rightarrow \{A\}$ holds on the union. This approach has also been applied by Wang and He for a similar use case [Wang and He, 2017]. As web tables from the same web site are expected to be less heterogeneous than web tables from different web sites, the union of all web tables from the same pay-level domain, called intra-PLD FDS, and the union of all web tables from different pay-level domains, called inter-PLD FDs, are created separately. A *pay-level domain (PLD)* refers to the part of a domain name that is paid for and is intended to capture the notion of a web site, i.e., all pages under the same PLD are assumed to belong to the same web site. The intra-PLD FDs are calculated on the union created from all web tables from the PLD, while inter-PLD FDs are calculated on the union of all web tables in the corpus.

To generate features for the classification model, different types of FDs are calculated on the union, which take the heterogeneity of web tables into account. Web tables from different sources might state the same information differently, for example, the population for Germany can be stated as 82.2 million in one web table and as 82 million in another. Instead of calculating hard FDs, which are violated as soon as two rows disagree on a value, probabilistic, or soft, FDs (pFD) and fuzzy probabilistic FDs (fpFD) are used here [Wang et al., 2009].

Probabilistic FDs state the probability of an FD being valid. The probability that a pFD $\{X\} \rightarrow \{A\}$ holds with respect to a specific entity V_X of subject column X depends on the number of rows with value V_X in the subject column, expressed as $|V_X|$, and the frequency of the most frequent value V_A of column A in these rows, expressed as $|V_A, V_X|$, in Equation 8.10.

$$Pr(X \rightarrow A, V_X) = \frac{|V_A, V_X|}{|V_X|} \quad (8.10)$$

The probability that the pFD holds for the whole table t is calculated as the average over all distinct subject column values D_X as shown in Equation 8.11.

$$Pr(X \rightarrow A, t) = \frac{\sum_{V_X \in D_X} |V_A, V_X|}{|D_X|} \quad (8.11)$$

A pFD with a probability close to 1 is an indication for a binary relation, while a probability close to 0 indicates an n-ary relation. Fuzzy probabilistic FDs are defined in the same way, but additionally use a similarity measure instead of checking value equality. For the feature generation, the same data-type specific similarity measures as in T2K Match (see Chapter 5) are used.

Features for Classification

Table 8.5 summarizes the features that are created from the FDs. Features (1) and (3) are the intra- and inter-PLD pFDs and feature (8) is the inter-PLD fpFD as described above. Feature (2) is the average over all intra-PLD pFDs, which describes the average probability of the dependency on all web sites. Features (4) and (5) set the inter-PLD pFD of the column in relation to the detected subject column and features (6) and (7) set it in relation to all other columns in the table. The pFD of an subject column is calculated using the correspondences to entities in the knowledge base and is a measure of the heterogeneity of the subject column's values, i.e., its probability is low if different values are mapped to the same entity. The intuition behind features (4) and (5) is that if the subject column's values result in a low pFD score, then the pFD score of other columns should not be expected to be higher. Feature (6) captures the general heterogeneity of all columns in the same web table and feature (7) measures if the column is more or less likely to be a binary relation than the other columns in the same web table.

Additional features are added as indicators for specific subcategories of independent columns. The detection of empty columns with feature (9) mainly involves removing encoding artefacts from the HTML-based representation, for example “ ”, which encodes a whitespace. Tables with columns of the list category, i.e., those that only list entities, are often sorted alphabetically either from left to right or top to bottom, which is captured by feature (10). Feature (11) is an indicator for horizontally stacked tables, i.e., multiple tables in the same HTML table tag, where the column headers are repeated in the exact same order and usually only empty columns are in between. Finally, feature (12) is an indicator for ranks, which often have the same column headers.

Evaluation

This section describes the training and evaluation of the supervised classifier for the proposed categories. The classification model uses the one-versus-rest strategy and linear regressions for each top-level category. For training and evaluation, the columns from the manually annotated sample are used. The training set consists of 192 tables with 699 columns and the test set contains the remaining 195 tables with 703 columns. To classify a column, all regressions are applied and the prediction with the highest score is used.

Table 8.6 shows the resulting confusion matrix. Overall, the model achieves an accuracy of 74.54% and precision and recall values between 67% and 86% for all classes. The features with the strongest impact for the binary and n-ary categories are (3)-(7) and for the independent category (9)-(11). As a baseline, a model that only uses feature (1) achieves precision and recall values of 49%/68% for binary and 58%/41% for n-ary.

Table 8.5: Features used for column classification.

#	Feature	Description
1	Intra-PLD pFD	The pFD calculated on all values from the same PLD
2	Avg. Intra-PLD pFD	The average of (1) over all PLDs that weighted by entity count
3	Inter-PLD pFD	The pFD calculated on values from multiple web sites
4	Subject Column Inter-PLD pFD	The (3) of the subject column of the table
5	Subject Column diff.	The difference between (3) and (4)
6	Avg. Table Inter-PLD pFD	The average of (3) of all columns in the table
7	Table Inter-PLD Ratio	The ratio between (3) and (6)
8	Fuzzy pFD	the fpFD of all values
9	Is Empty	1 if the column contains no characters except spaces
10	Is List	1 if all values in the table are sorted horizontally or vertically
11	Is Horizontally Stacked	1 if all column headers in the table are repeated in the same order
12	Is Rank	1 if the column header is “rank”, “#”, or “pos”

Table 8.6: Confusion matrix for the column category classifier.

	True Binary	True N-ary	True Ind.	Precision
Pred. Binary	219	72	16	71.34%
Pred. N-ary	74	252	9	75.22%
Pred. Ind.	2	6	53	86.89%
Recall	74.24%	76.36%	67.95%	

Table 8.7: Distribution of column categories in the WDC Web Tables Corpus 2012.

Category	Clusters	%	Columns	%
Binary	15 477	21.70%	585 074	25.90%
N-ary	44 192	61.90%	1 629 498	72.10%
Independent	11 691	16.40%	44 666	2.00%
Total	71 360	100.00%	2 259 238	100.00%

The confusion matrix shows that there is an equal amount of columns which are incorrectly classified between binary and n-ary. A reason for a misclassification of an n-ary relation as binary can be that there is not enough overlap among the entities in the merged web tables, resulting in a high probability for a binary relation. A misclassification of a binary relation as n-ary can be caused by very heterogeneous values, for example if different surface forms such as “*German*”, “*GER*”, and “*DE*” are used in different web tables. Matching errors also lead to incorrectly estimated dependency probabilities and can cause the same problems.

8.4.4 Corpus Profiling

This section presents the result of applying the column category classifier to the WTC 2012. The created profile shows that the majority of columns belongs to the n-ary category (62%) and only 22% are binary relations.

For the creation of the data profile, first all web table columns are grouped into clusters based on their class and property combination as determined by T2K Match. All web table columns which are not assigned to a property from the knowledge base, but which are in a web table that is assigned to a class from the knowledge base, are grouped by their column header to form clusters. These clusters are then used to calculate the features that were presented in the previous section.

Then, the classifier is applied to all individual web table columns with the feature values that are obtained from the clusters as well as the web table that contains the column. This step is limited to all columns which are in a cluster that contains at least two different web sites, which is required for the calculation of the inter-PLD pFD feature. The resulting dataset contains a total of 2 259 238 columns, which are grouped into 66 999 clusters.

Table 8.7 shows the distribution of categories for columns and clusters. Binary relations account for one fifth of all clusters (21.7%) and a quarter of all columns (25.9%), while the majority of both is classified as n-ary (61.9% for clusters and 72.1% for columns). The independent category is assigned to 16.4% of all clusters and 2.0% of all columns. The total amount of 71 360 category labels for the 66 999 clusters shows that not all columns of a cluster are classified with the same category. This can be the case if semantically similar attributes are dependent on additional information in some, but not all of the web tables in a cluster.

Table 8.8: Frequent attributes that are classified as binary relations.

Class	Attributes
BaseballPlayer	pos, hand, team, height, weight, no
Company	symbol, market cap, revenues, profits, assets, sales, employees
Country	sales tax rate, voltage, freq.
Device	resolution, colors, carrier, browsing time, music playback
EducationalInstitution	highest award, enrolled, state, top degree, website, % minorities, acceptance rate
Film	genre, year, (age) rating, studio, director, length
Fungus	shape, characteristic, hosts, context, lower surface, taste
RaceHorse	age, distance, track, division, sex, weight
Mineral	hardness, density, formula
Plant	color, life cycle, bloom season, bloom color, moisture, sun, vitamin a content, min temperature
Satellite	position, duration, mass, mission results, launch vehicle
Software	version, license, windows, platform, mac osx, latest stable

This result confirms the findings from the manual annotation and shows that the majority of columns in web tables do not comply with the data model that is assumed by current state-of-the-art approaches. This can be a reason for the low data quality that was observed for the extracted triples in Section 6.4: if the triples were extracted from columns that are in n-ary relations, then their values can be evaluated as false because the context, for example the point in time, is different from that in the knowledge base.

How n-ary relations can be used for schema extension will be further explored in Chapter 9. For the initial goal of this chapter, Table 8.8 shows examples of attributes for several DBpedia classes that were classified as binary relations. These attributes comply with the assumed data model and can directly be used to extend the schema of the knowledge base. All of the attribute ranking methods described in Section 8.3.2 can be applied to these results. The table shows the top-ranked attributes according to the conditional ranking method, i.e., the most frequent attributes for the respective classes.

8.5 Conclusion

This chapter analysed the potential of web tables for the schema-extension task. First, related work for attribute ranking and schema extension was introduced in Section 8.2. Then, Section 8.3 defined several measures for the ranking of attributes that are discovered from web tables and experimentally compared them in combination with schema matching techniques that map the schemata of web tables to each other. The results of these experiments showed that relevance is not a sufficient quality criterion for schema extension from web tables. Due to the frequent occurrences of n-ary relations in web tables, the completeness of new attributes must also be taken into account. Section 8.4 introduced a categorisation scheme for attributes that allows the judgement of completeness, described how a classifier for this categorisation was designed and evaluated, and profiled the Web Data Commons Web Tables Corpus 2012 according to the categories.

This chapter made the following contributions:

- **Attribute Ranking Methods:** Several attribute ranking methods were extended to consider correspondences between the web tables and a target knowledge base, created by T2K Match, and correspondences among the web tables, created by additional schema matching methods. These correspondences improved the estimation of co-occurrence frequencies, the main feature for attribute ranking, as attribute counts are no solely based on exact string matches of column headers.
- **Evaluation Methodology:** An experimental comparison of two evaluation methodologies for schema extension showed that the commonly used evaluation approaches based on relevance are problematic and can lead to inconclusive results. A qualitative analysis of highly ranked attributes for several ranking methods further revealed that many attributes cannot be understood without additional information. It is hence concluded that completeness is a necessary dimension for the judgement of the quality of new attributes for schema extension based on web tables.
- **Column Categorisation:** A categorisation scheme for the judgement of completeness of discovered attributes was defined and a classifier for this scheme was designed and evaluated. Through the application of this classifier, a data profile of the WTC 2012 according to this categorisation scheme was created, which shows that 62% of the discovered attributes represent a non-binary relation and are hence incomplete without additional data.

The experimental comparison of different ranking methods did not conclusively show a method that always outperforms all other methods. Depending on the used dataset and evaluation setting, different strategies were successful. However, it can be seen across all experiments that incorporating correspondences obtained from schema matching improves the results for all strategies. This finding

improves the state of the art, as most previous studies relied on exactly matching column headers when determining the frequencies of attributes in the web tables.

Concerning the evaluation of the ranking methods, a qualitative analysis of the results on the WTC 2012 showed that relevance alone is not sufficient to judge the quality of the newly discovered attributes. The attributes that are represented by single columns in web tables can often not be interpreted without additional information from other columns or the context of the web table, and are hence not suitable for the generation of interpretable triples for knowledge base augmentation. This leads to the conclusion that completeness needs to be considered as a further dimension of quality for schema extension based on web tables.

Completeness can only be achieved with the current methods if the new attributes are in a binary relation with the detected subject column of the web table. If an attribute is in an n -ary relation, additional data are required which are not available. Consequently, a categorisation scheme for columns, which distinguishes between columns that can be used to generate understandable triples and columns that need additional context in order to be understood, was introduced. Both, a manual empirical analysis as well as an automated large-scale experiment using a classification model, showed that the majority of columns in web tables belong to the latter category, i.e., need additional context for their interpretation.

This finding does not only show how to improve the results of current attribute ranking methods for knowledge base augmentation, but also indicates that the data model on which most semantic table interpretation methods are based is oversimplified. The common assumption, that the columns in web tables only represent binary relations is not true and so far, no method considers this. The implications of this finding will be further analysed in Chapter 9, which presents a method that considers n -ary relations and presents a detailed data profile of their occurrences in the WTC 2015.

Chapter 9

Synthesizing N-ary Relations from Web Tables

9.1 Introduction

The corpus profiling in Chapter 8 has shown that the majority of columns in the analysed web table corpus is in non-binary relations, i.e., that more than two columns need to be considered to understand their values. For example, a web table that states employment statistics for different states might do so by providing the total number of employees, or differentiate between different occupations and different points in time. This has implications for schema matching approaches as well as applications using the web table data. In this chapter, the focus is on the schema-extension task for knowledge base augmentation. When extending the schema of a knowledge base with n-ary relations, it is not possible to simply add a new property to the knowledge base and populate it with triples, as this would lead to the loss of explanatory values that are required for their interpretation. While there have been approaches to extract additional values related to time and units from web tables [Zhang and Chakrabarti, 2013, Oulabi and Bizer, 2017], until now no general-purpose method has been proposed that is able to detect and extract n-ary relations in from web tables.

This chapter introduces a method to synthesize n-ary relations from web tables. As web tables are mostly very small, it is necessary to synthesize larger tables through the use of stitching (see Chapter 7) before n-ary relations can be detected. Further, these tables need to be enriched with context information from the web page around the web tables, as the web tables often do not contain all explanatory attributes that are necessary to understand the values that are stated in them. These necessary attributes are identified by functional dependency discovery on the stitched tables and further enable the discovery of candidate keys as well as the normalisation of the tables.

After the introduction and evaluation of this method, it is applied to the WTC 2015 and the frequency, structure, and topical content of the synthesized n-ary relations is profiled. This profile verifies that a large fraction of the synthesized relations is non-binary and reveals that many seemingly equal attributes have different semantics based on the context in which they occur. This gives rise to the question how such data can be modelled in current knowledge bases, which is discussed with respect to three frequently discovered types of n-ary relations.

The contributions of this chapter are the following:

- **Method:** This chapter presents the first method to synthesize general n-ary relations from web tables. The method is able to exploit the page context around web tables in order to identify additional key elements. In addition to the identification of n-ary relations, the method extracts their values from multiple web tables and integrates them with the schema of a given knowledge base and is hence the first method that enables general schema extension with non-binary relations from web tables.
- **Data Profile:** The presented data profile is the first to consider binary relations and n-ary relations in a corpus of 5 million web tables with respect to key size and topical distribution, showing that web tables contain large amounts of n-ary relations. The data profile reveals that a large fraction of the data in web tables is of higher complexity than previously assumed and can only be interpreted if the context is considered. This context is often not part of the web tables, but needs to be identified on the web pages that contain the web tables.
- **Evaluation Dataset:** Evaluation datasets for the task of identifying n-ary relations in web tables are created and published, which are the first datasets for this task. These datasets consist of more than 300 thousand web tables and are considerably larger than previously published datasets for other tasks related to web tables. By publishing the datasets, the replicability of the results is ensured and a foundation for comparing methods that synthesize n-ary relations from web tables is provided.

The rest of this chapter is organized as follows: Section 9.2 compares the proposed method to related work. Section 9.3 gives an overview of the method and describes the different steps. Then, Section 9.4 presents an evaluation of the method and describes the datasets that are used for this evaluation. Finally, Section 9.5 presents the results of using the proposed method to profile a corpus of 5 million web tables.

The work presented in this chapter, i.e., the method to synthesize n-ary relations from web tables and their profiling, has previously been published in [Lehmberg and Bizer, 2019a, Lehmberg and Bizer, 2019b].

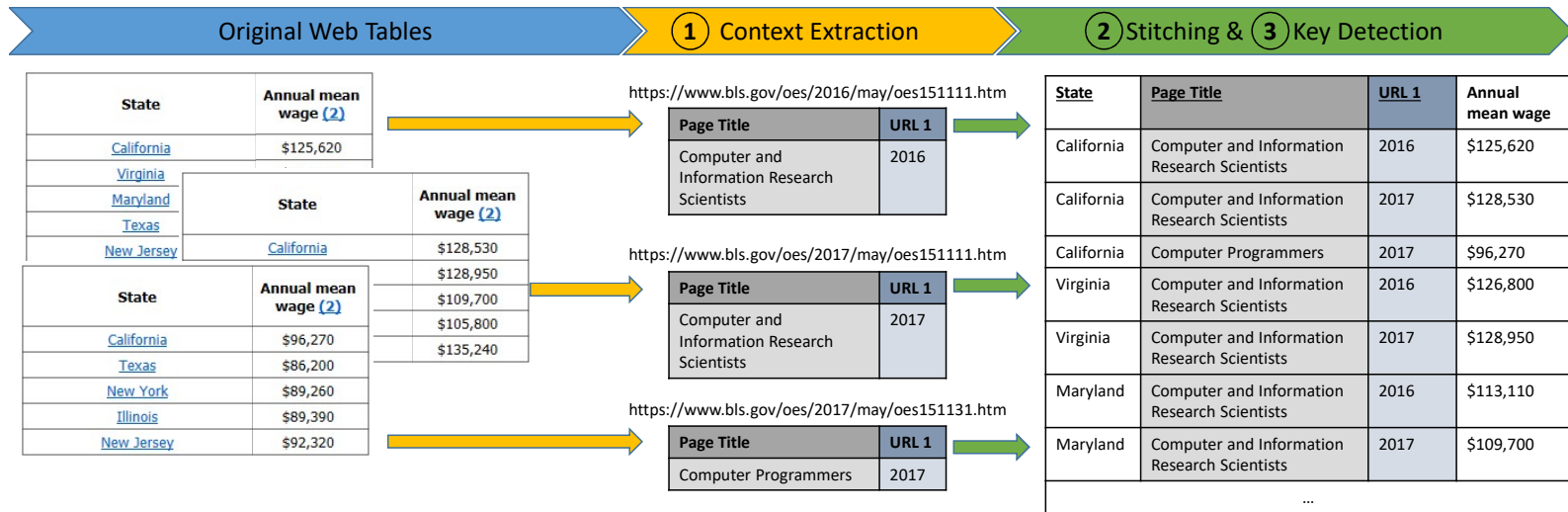


Figure 9.1: Overview of the applied method. The web tables of a single web site are extended with context attributes and stitched into larger relations, for which a key detection is performed to determine the set of attributes that can explain the values in the web table.

9.2 Related Work

This section briefly compares the literature in different areas that is related to the proposed method, such as table stitching, context extraction, and functional dependencies. A more detailed discussion of the literature on the relevant topics of semantic table interpretation (Chapter 5), table stitching (Chapter 7), and schema extension (Chapter 8) can be found in the respective chapters.

Context of Web Tables

The context of a web table consists of all the additional information that is available to a human user who sees the web table and can give important clues for its interpretation. This context comprises the textual content of the web page that contains the web table, but also its URL, which gives hints to the structural location of this page in the context of the respective web site.

The textual content surrounding web tables on a web page has been exploited by various approaches for the purpose of table understanding or search [Yakout et al., 2012, Zhang, 2017, Ritze and Bizer, 2017, Braunschweig et al., 2015], where the text surrounding a web table is used to calculate a contextual similarity among web tables or between a web table and a knowledge base class. Other approaches, such as the one proposed by Wang et al. [Wang et al., 2015b], determine which sentences on the same web page are related to a given web table. Only few approaches use table context to generate additional columns for the web tables [Ling et al., 2013, Cafarella et al., 2009], but even then, these columns are not further exploited by the methods and left to be interpreted by an end user. An exception are methods that use timestamp-metadata from the context of web tables for data fusion, i.e., the context data is used for the interpretation of specific columns and values in the web tables [Oulabi et al., 2016, Zhang and Chakrabarti, 2013].

Several other approaches have used the URLs of web pages as contextual information to find groups of web pages which are generated from the same template. Yin et al. [Yin et al., 2010] use a URL pattern summarizer to find web pages which are created from the same template and induce wrappers for these templates. A similar approach is described by [Song et al., 2015], who use the sets of web pages with common URL patterns for the extraction of entity names from their page titles. These approaches are similar to the URL-based clustering of web tables that is described in Section 9.3.2.

Functional Dependencies & Normalisation

There is a large body of work on efficiently discovering functional dependencies (FDs), and a recent comparison of several approaches is given by Papenbrock et al. [Papenbrock et al., 2015]. A functional dependency specifies a set of attributes that uniquely determines the value of another attribute (see Chapter 2). In this chapter, the TANE algorithm [Huhtala et al., 1999] is used for the discovery of

approximate functional dependencies and FD induction [Flach and Savnik, 1999] is used to combine the FDs that were discovered on different tables. Both are algorithms that discover FDs efficiently by applying pruning rules on their search space and hence do not have to check all possible combinations of attributes for all combinations of tuples, which has a worst-case runtime in $O(n^2(\frac{m^2}{2})2^m)$ in a relation instance with n tuples and m attributes [Liu et al., 2010].

Based on such functional dependencies, relational schemata can be normalised, which reduces redundancy and potential error sources. Algorithms for normalisation [Bernstein, 1976] assume that the functional dependencies, which they require as input, were created by an expert and only contain semantically meaningful FDs. When given FDs that were created by a functional dependency discovery algorithm, they might choose coincidental FDs, which hold on the specific relation instance that was used for discovery, but are not meaningful in general. This can lead to normalised schemata which are also not semantically meaningful. To prevent this, the decomposition algorithm proposed by Papenbrock and Naumann [Papenbrock and Naumann, 2017] uses heuristics to identify semantically meaningful FDs to perform normalisation based on FDs that are obtained through FD discovery.

Extraction of N-ary Relations

N-ary relations are essential for the representation of complex information and have been considered especially for the case of spatial and temporal data. For example, YAGO2 extends the frequently used model of RDF triples to 5-tuples, which contain a time and a location in addition to the usual subject, predicate, and object [Hoffart et al., 2013]. Wikidata further allows the annotation of triples with arbitrary properties that provide additional information [Vrandečić and Krötzsch, 2014]. A general overview of approaches how n-ary relations can be modelled in RDF is given by Hernandez et al. [Hernández et al., 2015].

The extraction of n-ary relations from unstructured text has been approached by several open information extraction systems from the field of natural language processing, such as Fact Extractor, FRED, Graphia, LODifier and Refractive [Augenstein et al., 2012, Exner and Nugues, 2014, Fossati et al., 2018, Freitas et al., 2012, Gangemi et al., 2017]. These systems use methods such as dependency parsing, discourse representation structures or frame semantics to extract n-ary relations from natural language sentences. As they rely on the natural language and sentence structure, which does not exist in tables, these methods are not applicable to web tables.

The idea of using dependency parsing for sentences is, however, similar to the usage of functional dependencies in the method presented in this chapter. The KrakeN system [Akbik and Löser, 2012], for example, uses the dependency parse tree of a sentence to navigate from a detected “*fact phase*”, which represents a relation, to all arguments of this phrase, which then become the arguments of the extracted n-ary relations.

https://www.bls.gov/oes/2016/may/oes151111.htm						
Generated context columns				Extracted web table		
Page Title	URI 1	URI 2	...	State	Employment (1)	Employment per thousand jobs
Computer and Information Research Scientists	2016	may	...	California	4,950	0.31
				Virginia	2,550	0.68
				Maryland	2,490	0.94
				Texas	1,810	0.15
				New Jersey	1,530	0.39
				Location quotient (9)	Hourly mean wage	Annual mean wage (2)

Figure 9.2: Example web table. The generated context columns shown next to the extracted web table are created from elements of the web page from which the web table was extract, such as the page title, URL, or heading above the table.

9.3 Method: SNoW

This section introduces the SNoW method (Synthesizing N-ary Relations from Web Tables). The goal of this method is to determine and extract the relations in which the columns in a web table participate. This entails not only recognising that a column is not in a binary relation, as demonstrated in Chapter 8, but also to discover which additional attributes participate in the relation, even if they are not explicitly stated in the web table.

To achieve this, the method first extracts additional data from the web page that contains the web tables into new “*context columns*”. These context columns are populated with the values extracted from the web page, i.e., constants, in the individual web tables. It is thus not possible to determine which of these columns are possible explanations for the original columns in the web table without additional data. Such additional data are included through table stitching (see Chapter 7), which creates the union of all web tables that use the same schema. As a result of the stitching operation, the context columns contain different values that were extracted from the various web pages containing the original web tables. Now, functional dependency discovery algorithms can be used to determine possible combinations of columns that uniquely determine the values in another column and hence provide an explanation for its values. The discovered functional dependencies hence identify relations that can be extracted from the web tables.

This process is based on the assumption that all web tables that originate from the same web site were generated by queries to the same database, which allows the reconstruction of a schema from which these web tables could have been generated. This has been shown to be reasonable for web tables from the same web site [Lehmberg and Bizer, 2017], hence, the method processes all web tables from a single web site at a time.

As an example, consider the web table in Figure 9.2, which contains employment statistics for a certain profession in several U.S. states at a given time. While a human can easily understand this, it is hard for an algorithm, as the table contains only very few examples (the figure shows the complete table) and important attributes, such as the profession and the date, are missing from the table. Existing

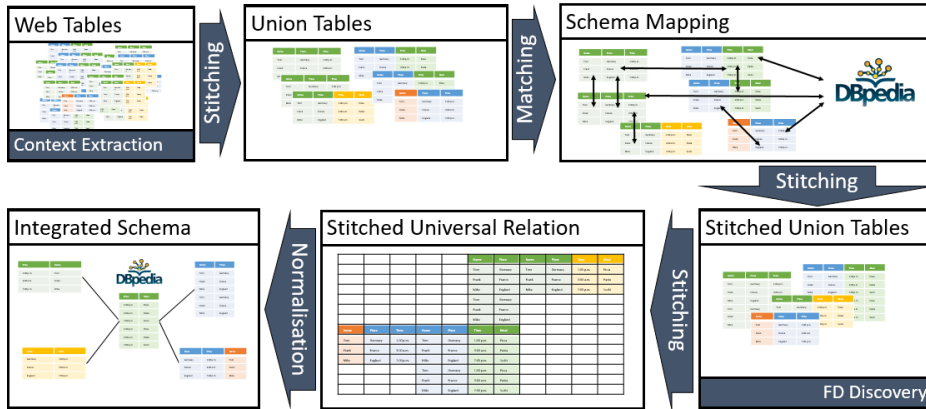


Figure 9.3: Overview of the SNoW method. The web tables of a single web site are first stitched into union tables, then into a universal relation, which is finally normalised into an integrated schema.

approaches use recognisers or matchers to find column pairs which represent an existing [Ritze et al., 2015, Dong et al., 2014a] or an unknown [Yakout et al., 2012] binary relation. In the example, such systems would extract a binary relation $\{state, employment\}$, which can be found in thousands of web tables from the same web site. As a result, the extracted binary relation contains about 2 000 different employment numbers for a single state and no possibility to choose a single, correct value, or to understand any individual value. To extract a meaningful relation for the *employment* attribute, all attributes which determine its value must be included, i.e., the functional dependency $\{state, page\ title, URI\ 1, URI\ 2\} \rightarrow \{employment\}$ must be discovered. This is only possible if additional data from the context of the web table is taken into account and if the data from multiple web tables, which contain employment numbers for different professions and dates, are combined.

9.3.1 Method Overview

This section gives an overview of the proposed method and briefly describes the individual steps, which will then be detailed in the following sections. The integration process that is executed by the SNoW method is visualised in Figure 9.3.

First, each web table is extended with context columns, which are generated from the web page that contains the web table. Then, web tables with identical schema are stitched into *union tables*. These union tables contain more rows than the original web tables and improve the performance of schema matchers (see Chapter 7), which are executed in the next step. The schema matchers create a mapping among the columns of the union tables, to identify matching schemata which are expressed with different column headers, and a mapping from the union tables to the knowledge base. Based on the mapping among the union tables, the stitching is repeated to merge all union tables with matching schemata into *stitched union ta-*

Computer and Information Research Scientists

<https://www.bls.gov/oes/2016/may/oes151111.htm>

Conduct research into fundamental computer and information science as theorists, designers, or inventors. Develop solutions to problems in the field of computer hardware and software.

Page Title	URI 1	URI 2	...	State	Employment (1)	Employment per thousand jobs	Location quotient (9)	Hourly mean wage	Annual mean wage (2)
Computer and Information Research Scientists	2016	may	...	California	4,950	0.31	1.64	\$60.39	\$125,620
				Virginia	2,550	0.68	3.59	\$60.96	\$126,800
				Maryland	2,490	0.94	4.99	\$54.38	\$113,110
				Texas	1,810	0.15	0.82	\$47.52	\$98,830
				New Jersey	1,530	0.39	2.04	\$60.52	\$125,880

Figure 9.4: Example for context extraction from the web page surrounding a web table.

bles. The stitched union tables contain the rows of all web tables with semantically matching schemata and enable the discovery of functional dependencies, which is infeasible on the much smaller original web tables. Finally, all stitched union tables which are mapped to the same knowledge base class are merged into a *universal relation*, which is normalised to create an integrated schema. This normalisation is guided by the mapping to the knowledge base and produces a star-shaped schema with the existing classes from the knowledge base at the centres and the n-ary relations that were discovered in the web tables referring to them. In the following sections, each step of this process is described in more detail.

9.3.2 Context Extraction

This section introduces the context extraction step, which creates additional *context columns* in the web tables based on data that are extracted from the web page that contains the web table and its URL. These additional columns are often necessary to explain the content of the original columns of the web tables. To ensure that the context columns that are created from the URL are aligned correctly for web tables from different web pages, a URL clustering step is used to find groups of URLs which are constructed from the same pattern.

Context Columns. Web tables are designed for human consumption and are always embedded in the context of a web page. Hence, not all data which are required to understand the content of a web table need to be explicitly stated inside the table, but additional values may also be stated elsewhere on the web page. To enable the method to make use of this context information, additional columns are created from the web page that contains a web table and certain value patterns in the web table are extracted into separate columns, called *context columns*. Specifically, all parts of the URL's path (split by '/'), the page title, and the heading closest to the web table are extracted into new columns. Further, numbers at the beginning of a cell (numbering columns) and values in brackets (disambiguation columns) are extracted into separate columns.

Figure 9.4 shows an example: The web table in the figure contains employment and wage statistics for different U.S. states. However, only when considering the context columns created from the page title and the URL, it becomes apparent that these statistics are for a certain profession (which can be determined from the page title) and from a certain date (which can be determined from the URL).

URL Clustering. Web sites may use different patterns to construct the URLs of their web pages and the same position in two URLs may have different semantics. For the extraction of additional columns from the URL, this means that the context columns of web tables from web pages with different URL patterns may have different semantics and should not be merged into the same column in a union table. To prevent this, a URL clustering step is applied that groups all URLs based on their most frequent element before creating context columns. The table stitching step is then only applied to all web tables in the same cluster, which prevents the merging of context columns with different semantics.

Consider, for example, the two URLs `/album/36grad` and `/artist/2raumwohnung`, where the second part (split by '/') refers to an album in the first URL and to an artist in the second URL. When creating the context columns, however, a new column "*URL 1*" is created for both URLs and applying the table stitching would merge these two columns with different semantics. When applying the URL clustering, one cluster contains all URLs starting with `/album/` and another cluster contains all URLs starting with `/artist/`. The web tables with these URLs are then only stitched within their respective clusters and the context columns with different semantics remain separated.

9.3.3 Schema Matching

This section introduces the schema matching component of the SNoW method. The first part, schema matching among union tables, re-uses the matcher introduced in Chapter 7. The second part, schema matching between union tables and the knowledge base, is similar to T2K Match as introduced in Chapter 5, but contains several changes to account for the specific challenges of matching stitched tables. As final step, the results of both matching steps are combined and inconsistencies are removed by a holistic global matching.

Table-to-Table Matching. Through the use of templates for the generation of web pages from a back-end database, many web tables on the same web site share the same attributes or even have exactly the same schema. However, this is not always obvious from the column headers in the tables, as they can use different languages or are simply non-existent. In these cases, a schema mapping containing correspondences among the columns of the union tables is needed to determine which columns represent the same attribute.

Such a schema mapping is created by applying the hybrid schema matcher introduced in Chapter 7, which combines instance-based and label-based schema matching with constraints derived from the schemata of the union tables. First, an instance-based schema matcher creates candidate correspondences by transforming the column contents into TF-IDF vectors and calculating their cosine similarity. This creates initial correspondences between the columns, which are then extended using a label-based matcher and refined by the constraint that no correspondence can hold between two columns which appear in the same union table. This constraint is based on the assumption that the same attribute does not appear more than once in the same web table and that the web tables from the same web site do not contain noise in the form of typos or other unsystematic differences as they are created from the same template. It corresponds to the graph-based refinement step described in Chapter 7 and is similar to the model proposed by Bronzi et al. [Bronzi et al., 2013] and has been successfully applied in several approaches [Lehmberg and Bizer, 2017, He et al., 2016, He and Chang, 2003].

Table-to-KB Matching. The second matcher aligns the union tables with the target knowledge base by identifying which classes and properties the input tables and their columns correspond to. This matcher is based on T2K Match, as described in Chapter 5, but introduces some modifications. First, candidate schema correspondences are generated using a computationally efficient matcher (Cosine similarity on TF-IDF vectors) as in the table-to-table matching step. These schema correspondences are then used to limit the set of candidate entities for each row, which are determined by a similar matcher as in T2K Match. These entity candidates are then used in a duplicate-based matcher to refine the initial schema correspondences.

These steps are conceptually the same as in T2K Match, with the difference that no upfront detection of a subject column is performed. This prevents that an incorrect decision during this detection step makes a correct schema matching of the table impossible. This means that, for the generation of candidate entities, all candidate schema correspondences between columns in the web table and classes in the knowledge base are considered.

Another modification is a de-duplication step for entity candidates before using them for duplicate-based schema matching. This ensures that each entity-value combination can only vote once during the matching, which prevents noise from the table extraction step and unequally distributed entities in the tables from dominating this matching step. While these are negligible effects when matching individual web tables, the stitching of potentially thousands of web tables can result in a very high weight for such errors or frequent entities during the voting stage, which is problematic if these frequent elements are not known in the knowledge base and lead to incorrect entity candidates.

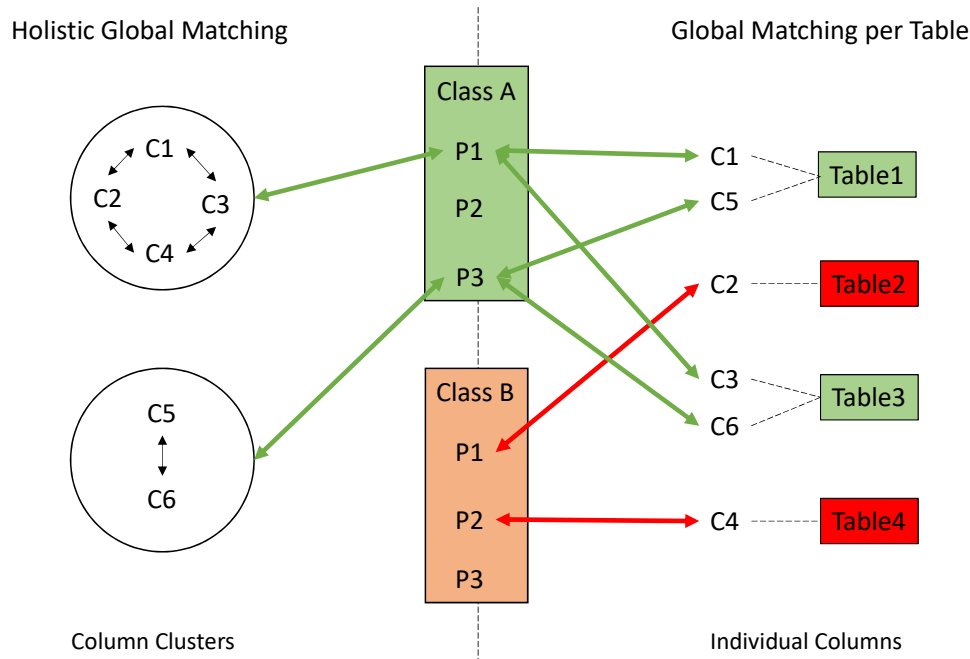


Figure 9.5: Holistic Global Matching: Columns are clustered before calculating a maximum weight bipartite matching.

Holistic Global Matching. After the duplicate-based schema matching, the result is a set of potential schema mappings for each union table to the knowledge base. At this point, T2K Match uses a top-1 global matching to decide for a final mapping. Here, however, the preceding table-to-table matching step provides additional information that can be exploited in a holistic global matching, as depicted in Figure 9.5. This step clusters the union table columns by calculating the connected components in the graph of table-to-table correspondences. Each cluster then describes one attribute in the web site's schema according to the table-to-table correspondences. Then, the scores of all table-to-KB correspondences are aggregated for each cluster, resulting in a similarity score between column clusters and properties in the knowledge base. Using these aggregated scores, a maximum weight bipartite matching decides for the mapping between the columns and properties holistically. This prevents inconsistencies between the correspondences among the columns in different union tables and their correspondences to properties in the knowledge base.

After applying these three matching steps, the created correspondences identify clusters of columns in the union tables which represent the same attribute in the web site's schema, and each of these attributes can have a correspondence to an existing property in the target knowledge base. As in the preceding chapters, only those union tables with correspondences to the knowledge base are considered, as otherwise, the content of the tables cannot be interpreted.

9.3.4 Schema Integration

To synthesize relations from the original web tables, the SNoW method performs several integration steps that merge and transform the schemata obtained from the web tables. These steps successively transform the schemata into a consolidated, universal schema that contains all attributes that are stated in the original web tables and their context. The following paragraphs discuss each of these integration steps.

Union Tables. The first step is to create union tables by stitching all web tables with the same schema. As in Chapter 7, a schema here refers to the ordered set of column headers of the web tables and includes the generated context columns. As mentioned in Section 9.3.2, context columns are created for clusters of web pages based on their URL pattern, and two context columns have the same column header only if they are created for the same cluster. This means that all context columns from different clusters are considered to represent different attributes. They can, in the same way as the original columns from the web tables, be matched during the schema matching phase.

Projection of existing Classes. The schema mapping between the union tables and the knowledge base identifies all known classes and properties in the union tables. To make the integrated schema consistent with these classes in the knowledge base, all columns in each union table that are mapped to properties which identify entities of the recognised class in the knowledge base are projected into a separate table and replaced with a foreign key in the union tables. This results in new tables which conform to the schema of the classes defined in the knowledge base.

Figure 9.6 shows an example. The union table in the top of the figure has been matched to the `Song` class and the columns “*Name*”, “*Album*” and “*Artist*” are marked as key attributes for this class. The projection step hence creates a new table for the `Song` class with these attributes, and replaces the corresponding columns by a foreign key in the union table, as shown on the bottom of the figure.

To bridge the differences between the relational model in the tables and the RDF data model in the knowledge base, the SNoW method uses candidate key definitions for each class in the knowledge base, which are specified as additional input of the method. These candidate keys can be a single property, such as `rdfs:label` for the class `Country`, or a combination of properties, such as `rdfs:label` and `musicalArtist` or `rdfs:label` and `album` for the class `Song`. The projected tables represent the schema overlap between the web tables and the knowledge base, and the union tables with foreign keys contain all potential candidates for schema extension.

Stitched Union Tables. A second stitching step creates the union of all union tables which have the same schema according to the schema mapping. This is an intermediate step to integrating all tables into a single schema, which is introduced specifically for functional dependency discovery: Union tables are only stitched if

Union Table

URI 0	Name	Album	Artist	Price	Time
GB	Love Will Tear Us Apart	The Best Of Joy Division	Joy Division	£ 0.99	03:26
AU	Love Will Tear Us Apart	The Best Of Joy Division	Joy Division	\$2.19	03:26
AT	Love Will Tear Us Apart	Nouvelle Vague	Nouvelle Vague	0,99 €	03:18
CA	Love Will Tear Us Apart	Lights Down Low	Damhnait Doyle	\$0.99	03:30
...					



FK	URI 0	Price	Time
1	GB	£ 0.99	03:26
1	AU	\$2.19	03:26
2	AT	0,99 €	03:18
3	CA	\$0.99	03:30
...			

PK	Title	Artist	Album
1	Love Will Tear Us Apart	Joy Division	The Best Of Joy Division
2	Love Will Tear Us Apart	Nouvelle Vague	Nouvelle Vague
3	Love Will Tear Us Apart	Damhnait Doyle	Lights Down Low
...			

Projected Tables

Figure 9.6: Example of projection of known classes.

all their attributes have correspondences, so this step does not introduce any null values. Null values require special treatment during functional dependency discovery, as they state that some information is unknown. As the very efficient, state-of-the-art FD discovery algorithms do not consider null values, this step stitches the union tables into the largest possible tables that can be created without introducing any null values. The details of functional dependency discovery and the effects of null values are described in Section 9.3.5.

Universal Relation. The final integration step is to stitch all stitched union tables into a single universal relation. In contrast to the previous step, all stitched union tables which are mapped to the same knowledge base class are merged and attributes which have no correspondences in some or even all other tables are populated with *null*-values. This corresponds to the minimum union operator as defined by Galindo-Legaria [Galindo-Legaria, 1994] (see Chapter 2), which first aligns the

schemata of the relation instances by adding missing attributes, which are filled with null values, creates the union, and then removes all subsumed tuples, i.e., those which have exactly matching values or missing values as indicated by the inserted null values.

Such a universal relation is the most general integration approach, as all data and their dependencies are preserved. Based on the schemata that are observed in the web tables, usually many different relational database schemata for the hypothetical database on the web server are possible. To avoid any decisions which could lead to the loss of information during the integration phase, all attributes are combined into a universal relation schema, which preserves all functional dependencies among the attributes. Based on this universal relation schema, the following steps, such as the normalisation step described in Section 9.3.6, applications, or end users can create a relational database schema that best suits their specific needs.

The stitching of web tables in several steps using the union and minimum union operators corresponds to a data fusion step, as defined in Chapter 2. However, the integration process used by the SNoW method does not apply any conflict handling, as all data conflicts, i.e., different attribute values for the same foreign key value, are used to discover functional dependencies among the attributes, which allows for the synthesis of n-ary relations. A conflict resolution at any stage in the process would remove the signals that enable this discovery and result in only binary relations between the entities that are represented by the foreign keys and the attributes in the universal relation.

9.3.5 Functional Dependency Discovery

This section describes the functional dependency discovery steps that are applied by the SNoW method to find potential sets of attributes which form a meaningful relation. In the following, the problematic of functional dependencies in the presence of null values is discussed and two functional dependency discovery steps that avoid it are introduced.

After extracting context columns and integrating the original web tables into stitched union tables as described in the previous section, all extractable attributes are known, but not how they are related to each other. Analysing the functional dependencies (FDs) that hold on the stitched union tables reveals which attributes depend on each other and must hence be synthesized into the same relation. A functional dependency $R : X \rightarrow Y$ states that for relational schema R , the values of the attribute set X uniquely (functionally) determine the values of the attribute set Y . This means, if a data source should be extended with an attribute from Y , then the attributes in X must also be added such that the values of the attribute in Y can be understood.

Functional Dependencies in Incomplete Relations

This section discusses FD discovery in the presence of null values and how the SNoW method handles such situations. As not all web tables contain the same set of attributes, stitching them into a universal relation as described in the previous section results in an incomplete relation instance. This means that attributes which are not contained in all web tables are padded with null values as their actual value was not observed and is unknown.

Under these conditions, Armstrong's axioms for standard FDs, which enable the reasoning over sets of FDs, do not apply anymore [Atzeni and Morfuni, 1986]. To still be able to reason over FDs, Levene and Loizou [Levene and Loizou, 1998] distinguish between strong $\Box(X \rightarrow Y)$ and weak $\Diamond(X \rightarrow Y)$ functional dependencies. A relation instance satisfies a strong FD if all possible interpretations, i.e., all possible replacements for the null values, satisfy the FD. A relation instance satisfies a weak FD, if there exists any interpretation that satisfies the FD. For strong FDs, the standard Armstrong axioms apply, but for weak FDs, the transitivity rule is changed such that it cannot hold via unknown values.

Table 9.1: An example of a table with null values.

A	B	C
a_1	\perp	c_1
a_1	\perp	c_2

An example is shown in Table 9.1, where the symbol \perp represents a null value. This table represents a relation instance which satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$ (based on a “no-information” interpretation of null values, i.e., two null values are not considered equal), but violates the FD $A \rightarrow C$ which follows from the other two by transitivity. According to Levene and Loizou, these are weak FDs $\Diamond(A \rightarrow B)$ and $\Diamond(B \rightarrow C)$, as there exists a possible replacement for the null values that satisfies the FDs, but not all possible replacements do so. Examples for strong FDs in this table are $\Box(C \rightarrow B)$ as well as $\Box(C \rightarrow A)$. In the context of the SNoW method, an FD $X \rightarrow Y$ is strong $\Box(X \rightarrow Y)$ if and only if all original web tables which contain Y also contain X .

Most algorithms for functional dependency discovery, however, are not designed to handle null values and treat them like a regular value, i.e., two null values are considered equal. So, to prevent that the stitching procedure introduces null values which might change the result of the functional dependency discovery algorithm, the stitched union tables are created such that no null values are introduced. This makes it possible to use any functional dependency discovery algorithm and benefit from the many efficient implementations that have been proposed in the literature. Missing values in the original web tables, i.e., empty cells, are treated as regular values.

Approximate Functional Dependency Discovery

The SNoW method applies functional dependency discovery to the stitched union tables. These tables may contain errors introduced during the extraction of the web tables, errors introduced by an incorrect schema mapping as well as inconsistencies such as misspellings that exist in the original database from which the web tables were generated. To account for these possible errors, approximate functional dependencies are used. Approximate functional dependencies (also known as soft FDs) are FDs which can be violated by a certain amount of tuples, which is specified by the approximation rate. Given a relation instance r , the functional dependency $r : X \rightarrow Y$, and the set of tuples v that violate the dependency, the approximation rate is defined as in Equation 9.1. For the discovery of such approximate FDs, the TANE algorithm [Huhtala et al., 1999] with an approximation rate of at least 0.95 is run on the stitched union tables.

$$\text{ApproximationRate}(r : X \rightarrow Y) = 1 - \frac{|v|}{|r[X \cup Y]|} \quad (9.1)$$

The result of applying the TANE algorithm is a set of approximate functional dependencies for every stitched union table. The approximation rate is only considered during discovery and in the following steps all approximate FDs are treated like regular FDs. The discovered FDs describe the functional relationship among the columns in the stitched union tables based on the data in their rows. This means that a discovered FD $X \rightarrow Y$ can be under-estimated, i.e., the set X is missing attributes, and that different FDs can be discovered for two stitched union tables which share common subsets of their attributes. Under-estimation of FDs happens if the table does not contain any data conflicts that indicate that additional explanatory attributes must be added to the determinant and often happens if the tables only contain few rows.

Functional Dependency Stitching

After the discovery of functional dependencies on the stitched union tables, the SNoW method integrates these tables into a universal relation as described in Section 9.3.4. During this process, the functional dependencies that were discovered on the stitched union tables must be merged, too. Merging functional dependencies that were discovered on different tables means to infer the FDs that hold for a common relational schema based on the observations made from different relation instances. This cannot be achieved by the union of the sets of FDs that were discovered on the relation instances, as an FD that is satisfied by one relation instance might be violated by another.

The functional dependencies that hold for a universal relation are obtained by merging the functional dependencies that were discovered on the stitched union tables in a way such that the resulting FDs are satisfied by all stitched union tables. This is achieved by selecting the most specific FDs from all sets of discovered FDs.

$$\{State\} \rightarrow \{Annual\ mean\ wage\}$$

<u>State</u>	Page Title	URL 1	Annual mean wage
California	Computer and Information Research Scientists	2016	\$125,620
Virginia	Computer and Information Research Scientists	2016	\$126,800
Maryland	Computer and Information Research Scientists	2016	\$113,110

(a)

$$\{State, Page\ Title, URL\ 1\} \rightarrow \{Annual\ mean\ wage\}$$

<u>State</u>	<u>Page Title</u>	<u>URL 1</u>	Annual mean wage
California	Computer and Information Research Scientists	2016	\$125,620
California	Computer and Information Research Scientists	2017	\$128,530
California	Computer Programmers	2017	\$96,270

(b)

Figure 9.7: Example of two tables with the same schema that satisfy different sets of functional dependencies.

The same approach is taken by state-of-the-art FD discovery algorithms that draw samples from large relation instances and then combine the FDs that are discovered on these samples [Papenbrock and Naumann, 2016].

As an example, consider the tables shown in Figure 9.7: As long as no pair of tuples for one of the mentioned states (column “*State*”) with a different profession (column “*Page Title*”) or year (column “*URL 1*”) is observed, it is discovered that the “*Annual mean wage*” column only depends on the state, as shown in Figure 9.7a. However, if the combination of state, profession, and year is observed to be necessary to uniquely determine the wage in another table, as shown in Figure 9.7b, then it can be inferred that this FD holds for every table which contain these attributes.

The algorithm that is used to merge the FDs for the universal relation is functional dependency induction [Flach and Savnik, 1999]. Instead of creating the union of the sets of functional dependencies that are the result of the TANE algorithm, the union of the complementary sets of non-FDs is created. This is also called a negative cover of FDs, and is the input that is required for the functional dependency induction algorithm proposed by Flach and Savnik. The algorithm then induces the most specific functional dependencies which are not invalidated by this negative cover.

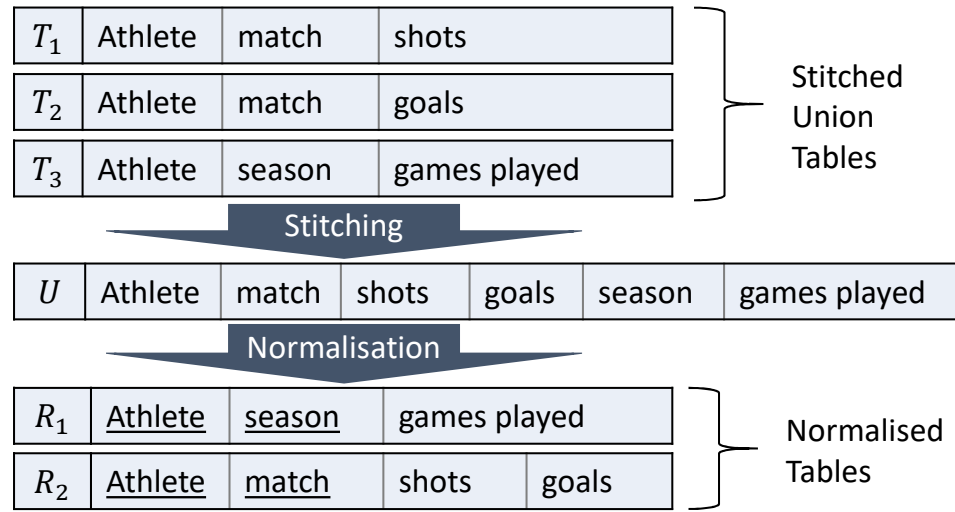


Figure 9.8: Example of the normalisation process.

The result of applying this procedure is a set of functional dependencies for the universal relation that is satisfied by all stitched union tables, and hence also by all original web tables. This means that these FDs are the most specific ones that were observed in any of the stitched union tables and describe the functional relationship among the attributes of the universal relation. This makes them candidates for semantically meaningful relations, which can be used to extend other data sources such as a knowledge base.

9.3.6 Schema Normalisation

This section describes how the universal relation that is the result of the table stitching steps is decomposed into a normalised relational schema. This reduces the sparsity of the data and makes it easier for end users to understand the schema. However, it is not an obligatory step and applications may choose to skip it, depending on the specific use case.

A universal relation contains all attributes that are stated in the original web tables which are mapped to the same class in the target knowledge base as well as possibly multiple FDs for each of these attributes. While this form preserves all possible dependencies among the attributes, it is not easily understood by human users and can result in a relation instance with many missing values, so applications might be interested in obtaining a normalised schema. To create normalised relations from the universal relation, it is desirable to select meaningful FDs for each attribute and project attributes with the same determinant into the same relation. This is achieved by applying a normalisation algorithm to the universal relation.

For example, consider the universal relation U for the class *Athlete*, as shown in Figure 9.8, where *Athlete* indicates the foreign key for the *Athlete* class, and the FDs $\{Athlete, match\} \rightarrow \{shots, goals\}$ as well as $\{Athlete, season\} \rightarrow \{games$

played}. Based on these FDs, the universal relation can be normalised into the relations $\{Athlete, match, shots, goals\}$ and $\{Athlete, season, games played\}$. The challenge for this normalisation is to choose meaningful FDs from the set of FDs that was created by the functional dependency discovery step. As the discovered FDs are only based on the data values and not on a semantic understanding, coincidental FDs such as $\{Athlete, shots\} \rightarrow \{goals\}$ might also be discovered. These are undesirable for normalisation, as they do not hold for all possible relation instances and do not represent a semantic relationship between the attributes.

To create a normalised schema from a universal relation, the decomposition algorithm proposed by Papenbrock and Naumann [Papenbrock and Naumann, 2017] is applied, which acknowledges the existence of coincidental FDs obtained through FD discovery. The algorithm requires a relation R and its extended functional dependencies as input. An extended FD is an FD $X \rightarrow Y$ that has its dependant extended to its closure, i.e., the set Y contains all attributes that transitively depend on X , where X^+ denotes the transitive closure: $Y = X^+ \setminus X$. The algorithm scores all FDs which violate the desired normal form and then selects the violation $X \rightarrow Y$ with the highest score to decompose the relation into two decompositions $R_1 = X \cup Y$ and $R_2 = R \setminus Y$. Then, the same is applied to each of the created decompositions.

To preserve the direct relation with the detected class from the knowledge base, only FDs which contain the class' foreign key in their determinant are considered and the algorithm is stopped as soon as all relations are in 2NF. The second normal form (2NF) specifies that no non-key attribute in a relation may be functionally dependent on any proper subset of any candidate key of the relation [Codd, 1972]. This normal form is suitable, as it reduces the redundancy that is present in the universal relation, but does not decompose it to a point where attributes might no longer be in a direct relation with the detect class from the knowledge base.

Violation Selection

This section introduces the scoring heuristics that are used to select violating FDs during normalisation. The selected FDs should be semantically meaningful as they define the schemata of the normalised relations. To achieve a semantically meaningful normalisation, the algorithm scores each violating FD $X \rightarrow Y$ using several indicators and chooses the violation with the highest score for decomposition. For example, the coincidental FD $\{Athlete, shots\} \rightarrow \{goals\}$ from the example above should receive a low score, as the *shots* attribute is not a meaningful determinant for the *goals* attribute and would lead to a normalised relation that does not contain any information about the match in which these goals were scored. The scores are designed to prefer determinants which have few attributes, contain attributes that occur in many original web tables, and provide interpretable values such as strings or dates rather than numbers. The violation score is the mean of the following individual scores:

Length Score. Assigns a higher score to FDs with few attributes in the determinant and many attributes in the dependant, see Equation 9.2, as coincidental FDs tend to contain more attributes in the determinant [Papenbrock and Naumann, 2017].

$$\text{Score}_{\text{length}} = \frac{1}{2} \left(\frac{1}{|X|} + \frac{|Y|}{|R| - 2} \right) \quad (9.2)$$

Data Type Score. Meaningful determinants should contain names or temporal references, hence a score of 1 is assigned if an attribute has either the type `string` or `date`, as shown in Equation 9.3. Numeric attributes in web tables usually contain factual data and are hence not considered semantically meaningful. This is different from a single database setting, where unique integer values are often used as keys.

$$\text{Score}_{\text{type}} = \frac{1}{|X|} \sum_{a \in X} \begin{cases} 1 & \text{if } \text{type}(a) \in \{\text{string}, \text{date}\} \\ 0 & \text{otherwise} \end{cases} \quad (9.3)$$

Provenance Score. The fraction of rows which are stitched from original web tables that contain the determinant attributes. The intuition is that meaningful determinants should be found consistently in many tables, while coincidental ones consist of attributes that only occur in fewer tables. In Equation 9.4, \mathcal{T} refers to the set of all web tables that were stitched into the universal relation, and \mathcal{T}_X refers to all of these web tables which contain the attributes in X .

$$\text{Score}_{\text{provenance}} = \frac{\sum_{t \in \mathcal{T}_X} |t|}{\sum_{t \in \mathcal{T}} |t|} \quad (9.4)$$

Entity Disambiguation Score. Set to 1 if the determinant contains a disambiguation attribute (see Section 9.3.2) that was generated for an attribute of the class to which the table is mapped, otherwise 0.

Disambiguation Split Score. Set to 0 if the FD contains a disambiguation attribute, but not the attribute which is disambiguated.

Filename Score. Set to 0 if the content of an attribute in the determinant is recognised as a filename, otherwise 1 (relevant for context columns created from the URL, which might contain the file name of the web page).

The heuristics proposed above are rather generic and serve as a basic means of selecting meaningful FDs for knowledge base augmentation. Applications which have a more specific task might extend or replace these heuristics to influence the normalisation procedure with respect to their specific goals. However, this has not yet received attention by the research community and is the scope of future work.

9.4 Evaluation

This section presents an evaluation of the SNoW method. The method is evaluated with datasets created from several web sites in the WTC 2015 with respect to the quality of context extraction, schema matching, and relation synthesis. The quality of relation synthesis is measured with respect to the completeness and conciseness of the synthesized relations.

9.4.1 Evaluation Datasets

To evaluate the method, several datasets are annotated, each containing all web tables from a single web site, with a total of more than 300 thousand web tables. These datasets are created from the same subset of the WTC 2015 as in Chapter 7. The selected web sites are among the largest websites in the used web tables corpus that are known to have overlap with the knowledge base, and were chosen to cover a variety of different topics.

The annotations comprise the schema mapping among the columns of all union tables from the same web site, the schema mapping from the columns of the union tables to the properties of the target knowledge base, and functional dependencies among the attributes represented in the union tables. As the schemata of web tables are not necessarily normalised, it is not feasible to assign annotations “*binary*”, “*ternary*”, etc. to a schema as a whole. It is rather necessary to identify a set of functional dependencies which specify the relations that are represented by the schemata of the web tables. Functional dependencies are annotated by running the TANE [Huhtala et al., 1999] FD discovery algorithm on all union tables, manually correcting the results based on the knowledge about the schemata and general background knowledge by adding or removing attributes, and then selecting meaningful FDs for each attribute.

Table 9.2 gives an overview of the datasets. The first two columns show the source web site and the number of web tables. The next three columns show statistics about the created union tables, the number of original columns (column “ A_O ”), and the total number of columns including context columns (column “ A_T ”). The last two columns show statistics about the created annotations. Column “ A_U ” shows the number of attributes in the universal relation for the respective web site, and column “ FDs ” indicates the number of annotated FDs for these attributes. These datasets contain data about athletes and their performance (d3football), famous people (nndb), statistics about countries and states (bls, cia), flights for different airlines (flightaware), events and their locations (seatgeek), music stores (itunes, amoeba), as well as charts and release information for video games (vgchartz).

To reduce the otherwise infeasible workload of annotating more than 300 thousand web tables, the fact that web tables from the same web site often re-use the same schema and can be merged into union tables [Ling et al., 2013] is exploited. This reduces the workload to the manual annotation of about 500 union tables. To

Table 9.2: Annotated Dataset Statistics. A_O =original attributes, A_T =total attributes (original & context attributes), A_U =universal attributes.

Web site	Web Tables	Union Tables			Annotation	
		Tables	A_O	A_T	A_U	FDs
d3football.com	40 584	12	63	145	41	18
data.bls.gov	10 824	12	61	181	51	12
flightaware.com	2 888	6	22	72	35	13
itunes.apple.com	42 729	76	470	1 095	59	6
seatgeek.com	157 578	72	266	714	71	30
www.amoeba.com	5 529	65	227	712	42	13
www.cia.gov	30 569	213	562	2 225	323	162
www.nndb.com	23 522	29	123	299	29	10
www.vgchartz.com	23 258	8	39	87	36	13
Sum	337 481	493	1 833	5 530	687	277

verify the assumption that this results in valid annotations, 10 randomly selected original web tables for each union table (or all if the table was stitched from fewer web tables) are checked manually, which is the same methodology as used by Ling et al. [Ling et al., 2013]. This shows that 4% of the created union tables violate the assumption (they contain columns which are labelled with a symbol, such as “%”, and the actual attribute name is stated outside of the web table). These incorrect columns are excluded from the annotations.

Table 9.3 shows examples of functional dependencies from the annotations. The examples in the table show that a broad range of relations with different arities is described by the annotated functional dependencies. This includes relations which are modelled in a way that their data can be represented as triples, i.e., the determinant specifies an single entity. These are the relations for which all existing methods are applicable. But, it also contains examples which contain additional information in their determinants, such as other entities or temporal references. The data in such relations cannot be processed properly with the existing methods. Over all datasets, 62.5% of the attributes in the annotations are dependent on at least one context attribute. All datasets are published to ensure the repeatability of the results as well as to enable the comparison of methods for identifying n-ary relations.¹

Target Knowledge Base

The used target knowledge base is again DBpedia. Knowledge bases such as DBpedia do not explicitly define keys for their classes, which specify the set of attributes that uniquely identifies an entity, as required for the schema integration

¹<https://github.com/olehmborg/snow>

Table 9.3: Examples of FDs in the annotated datasets. Attributes marked with * are context attributes, attributes marked with ** consist of original and context attributes.

Class	Determinant	Dependant
AdministrativeRegion	state	state code
AdministrativeRegion	state, profession*, year*, month*	employment, annual mean wage
Airline	airline, from/to*	cargo weight
Airline	airline, from/to*, routing, operated by*	popularity
Athlete	name, match*, action*	yds, avg, pts
Company	name	founded, employees
Country	country, date	population, gdp
Country	country	climate, geo. coord.
OfficeHolder	name, since	district
Person	name, birth date	occupation, death
Single	song, artist**, version*, album**, album edition*	runtime
VideoGame	name, region, platform*	release date, publisher

part of the SNoW method (see Section 9.3.4). Hence, a manually curated version of DBpedia is created by selecting classes for which candidate keys are explicitly defined. While the `rdfs:label` property is a sufficient key for most classes, others such as `Single` require additional attributes, for example `musicalArtist`.

Although the values of the `rdfs:label` property in DBpedia are unique, these values are often not sufficiently informative to uniquely identify an entity. For example, the labels of two entities with the same name might be disambiguated by adding the class name or the value of another property, such as the artist's name for songs, in brackets. These disambiguations are, however, not created consistently and the used property values might not be stated in the web tables, which creates ambiguity during the matching steps. This ambiguity is resolved by manually specifying the possible additional properties as candidate keys and removing the disambiguation parts from the `rdfs:label` values.

When defining candidate keys, the semantics, uniqueness and density of each property defined for a class are taken into account. The curated version of DBpedia contains 20 classes from the largest high-level classes `Organisation`, `Person`, `Place`, `Work` and their sub-classes in the DBpedia type hierarchy, which reflect the most frequently found classes in the used web tables (see Chapter 6), with a total of 2 369 349 entities.

Context Extraction

To evaluate the URL clustering step as described in Section 9.3.2, all URLs in the datasets are manually grouped based on their underlying pattern and compared to the result of the proposed method. The evaluation shows that only 172 out of 180 095 URLs are assigned to a wrong cluster, resulting in 195 web tables being merged into the wrong union table. Given the total of 337 481 web tables, the error rate is only 0.05%, which shows that the URL clustering correctly identifies the URL patterns for almost all URLs.

9.4.2 Schema Matching

This section discusses the evaluation of the schema matching steps of the SNoW method. The schema matchers are evaluated using the annotations for the columns in the evaluation datasets. These annotations contain an assignment of every union table column to a class in the knowledge base and to an attribute of the universal relation for the respective web site (column A_U in Table 9.2), including attributes which do not exist in the knowledge base. For comparability with other approaches, correspondences are created among all original web table columns which are annotated with the same attribute and the performance of the schema matchers is evaluated against these correspondences.

The column “*Schema*” in Table 9.4 shows the achieved F1-measure for each dataset. In many cases, the performance of the schema matcher is close to 100%, which is for a large part due to the very regular structure of the web tables. However, a detailed error analysis shows that the schema matchers do not handle certain numeric attributes very well, which affects the *d3football*, *bls*, *itunes* and *cia* datasets. Closer inspection of these attributes reveals that they have very similar domains and are barely distinguishable for an instance-based matcher. In addition to that, some of them also have no column headers, so there are no signals the matchers can exploit. One example is the first column in every web table in the *itunes* dataset, which has no column header and can either be the track number of a song on a specific album or just a row number for tables which contain songs from multiple albums.

This evaluation shows that the schema matching task within the scope of a single web site can be solved with very high quality using the methods proposed in this chapter. The main factors for this high quality are the stitching of individual tables, as already shown in Chapter 7, and the holistic global matching that was introduced in Section 9.3.3, which further improves the quality of the results.

9.4.3 Relation Synthesis

This section evaluates the SNoW method with respect to quality of the synthesized relations. Specifically, the completeness and conciseness of these relations is evaluated. This evaluation is based on the argument made in Chapter 8, where it was found that the evaluation of existing schema-extension methods, which is only

Table 9.4: Experimental results for schema matching and relation synthesis (F1-measure).

Web site	Schema	Binary	N-Ary	N-ary +C	Match	Norm
d3football.com	0.779	0.495	0.480	0.851	0.765	0.765
data.bls.gov	0.895	0.415	0.485	0.984	0.900	0.900
flightaware.com	0.982	0.459	0.405	1.000	0.942	0.733
itunes.apple.com	0.827	0.680	0.674	0.953	0.809	0.763
seatgeek.com	0.999	0.988	0.992	0.984	0.962	0.962
www.amoeba.com	0.933	0.812	0.394	0.907	0.885	0.885
www.cia.gov	0.976	0.858	0.836	0.821	0.660	0.635
www.nndb.com	1.000	1.000	1.000	1.000	0.999	0.999
www.vgchartz.com	1.000	0.448	0.253	1.000	1.000	1.000
Macro Avg.	0.932	0.684	0.613	0.944	0.880	0.849

based on relevance, is insufficient as it does not consider the completeness of the discovered attributes. The SNoW method addresses exactly this critique by synthesizing n-ary relations which contain additional explanatory attributes and hence represent complete relations for schema extension in the sense that no additional information is required to understand the data. To ensure that this does not lead to the inclusion of extraneous attributes, the conciseness of the synthesized relations is evaluated, too.

To evaluate the synthesized relations, one reference relation for each FD in the annotations is created and populated with the values from all web tables based on the manually created schema mapping. Then, the same methodology as used by Wang et al. [Wang and He, 2017] is applied: for each reference relation, the FD from a synthesized relation which is the best match is chosen, but each FD can only be evaluated for one reference relation. Then, the cell-wise overlap between the reference relation and the synthesized relation is calculated.

Consider the example relation $\{match, player, goals\}$ with four tuples. If the correct FD $\{match, player\} \rightarrow \{goals\}$ was detected, the synthesized relation contains 12 correct values (3 values for each of the 4 tuples). However, if the (under-estimated) FD $\{player\} \rightarrow \{goals\}$ was detected, only 8 of these values are contained, resulting in a recall of $\frac{8}{12} = 0.67$. If the FD was over-estimated and contains one additional, incorrect attribute in its determinant, the synthesized relation contains 16 values (assuming no additional tuples were created), resulting in a precision of $\frac{12}{16} = 0.75$.

This evaluation methodology takes the amount of values for each synthesized relation into account and is hence more meaningful for measuring completeness than an evaluation that only considers the schema level. Such an evaluation could measure which of the FDs in the annotations were correctly discovered, for example by checking if an FD from the annotations is both satisfied and minimal for a

given synthesized relation. But as the distribution of web tables over schemata is such that most web tables share very few schemata, a few web tables with a different schema, which can be considered outliers, would dominate a schema-level performance measure. For example, the most frequent schema for the `data.bls.gov` web site is used by 5 105 web tables and the least frequent schema is used by a single web table. However, both schemata each contribute 4 FDs to the annotations. If the evaluation was only performed on the schema level, each of these FDs would have the same influence on the result. With the goal of synthesizing as many correct and complete tuples as possible, however, it is more meaningful to assign a higher importance to the FDs for the frequently used schemata, which result in the larger set of tuples.

To evaluate the influence of each of the different components that are used by the SNoW method on the results, the following discusses the results of running the method in several different configurations. Some of these configurations use the schema mapping and FDs that are created by the human annotators, while others use the schema matcher and FD discovery algorithms. The comparison of these different configurations hence also shows the performance gains that can be expected in scenarios where user interaction with the system is possible. Figure 9.9 shows an overview of the results for these configurations.

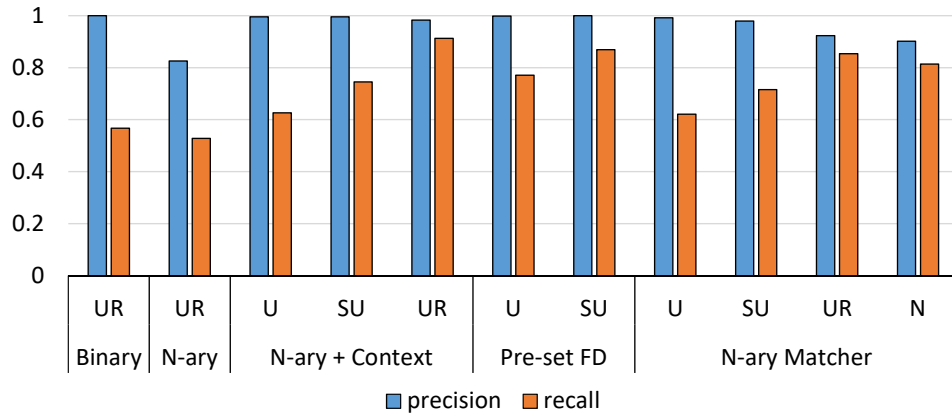


Figure 9.9: Results of the relation synthesis experiments. *U* = Union Tables, *SU* = Stitched Union Tables, *UR* = Stitched Universal Relation, *N* = Normalised Relations.

Configurations without matching

The following configurations load the schema mapping from the annotations instead of using the schema matching component. This allows for the analysis the method's performance without the influence of errors which are introduced during matching and represents possible configurations that can be used if the system is applied to only a small number of web sites.

Binary. The argument for the extraction of n-ary relations is that binary relations are incomplete. The extent of this incompleteness is shown by the result of this configuration, which resembles a perfect extractor for binary relations. This configuration uses the manually annotated schema mapping as input and extracts a binary relation for each reference relation. Averaged over all datasets, this configuration reaches a recall of 56.7% and a precision of 100% (binary relations cannot contain extraneous attributes). The low recall shows that most of the information is ignored with this approach. The incompletely synthesized relations contain only 56.7% of the values of the expected result. The precision of 100% is explained by the fact that conciseness is only of concern if additional attributes are included, which is not the case for this configuration.

N-ary. The next configuration uses functional dependency discovery instead of binary relations, but skips the creation of context columns, i.e., only the data that is available in the original web tables is used. On average, this configuration achieves 48.6% recall and 83.2% precision. The low precision and recall are explained by incorrect attributes that are included in the synthesized relations due to the lack of a better alternative, i.e., the explanatory attributes in the reference relations do not exist in the original web tables. The low recall is further caused by cases where no functional dependencies could be discovered (again, due to explanatory attributes not being in the original web tables), which means that all existing attributes are synthesized into a single relation. As the evaluation methodology states that every synthesized relation can only be evaluated for a single reference relation, not all reference relations are counted as successfully extracted and the recall is even lower than in the binary configuration. These problems lead to an overall worse performance than in the binary configuration and show that a naïve approach to the extraction of n-ary relations results in no improvement over a binary extractor. This highlights the importance of context extraction, which is often essential for the understanding of the content in the web tables.

N-ary + Context. The configuration using functional dependency discovery with context columns shows the optimal output of the SNoW method with an average recall of 92.2% and average precision of 98.1%. These high performance values show that (1) given enough data and a correct schema mapping, the task is solvable with the proposed method and that (2) the extraction of context columns plays a critical role, as the comparison with the previous configuration shows that the original web tables do not contain enough attributes to explain their data. The achieved recall is not perfect due to the under-estimation of some FDs, which indicates that the collection of additional web tables from the respective web sites could improve the result. The small amount of errors that influence precision is due to inconsistencies in the data that could not be mitigated by the use of approximate FDs.

Pre-set FDs. In this configuration, all functional dependencies and the schema mapping are loaded from the annotations. By measuring the performance after the first two stitching steps, the impact of each stitching step on recall can be determined. This configuration results in a recall of 86.9% for the stitched union tables and 77.1% for the union tables. This means that after the first stitching step, 77.1% of all values are already contained in the largest union table. The second stitching step increases this to 86.9%, and the remaining 13.1% are added when merging all stitched union tables into the universal relation. This is not surprising, as the profiling in Chapter 7 indicated that the distribution of web tables per schema is heavy-tailed, i.e., a small number of schemata is used for the majority of web tables. This means, a large fraction of all web tables uses the same schema and is merged into the same union table, which then contains the majority of the extractable data.

The experiments presented so far show that the discovery of n-ary relations in web tables depends on the extraction of additional data from the context of the tables. Further, the importance of stitching is shown by the increased recall after each stitching step.

Configurations with matching

In the preceding experiments, the performance under perfect conditions was evaluated, i.e., the schema mapping was known. In a large-scale use case, however, such annotations are not available and the schema matching methods are required to calculate this mapping. The following configurations use the schema matching component and hence show the end-to-end performance of the method for a large-scale scenario. The results of these configurations are expected to be worse than the results presented above, as the result of schema matching can contain errors and hence introduce noise into the results.

Matcher. This configuration does not load any annotations and uses the schema matching as well as the functional dependency discovery components. In this configuration, the method achieves an average recall of 87.0% and an average precision of 92.5%. Compared to the configuration “*N-ary + Context*”, this shows a drop in recall of 5.2 percentage points and a drop in precision of 5.6 percentage points. There are multiple reasons for the drop in both precision and recall: (1) missed correspondences result in missing values and under-estimated FDs, as not all columns that represent the same attribute in the universal relation are merged, affecting recall; and (2) incorrect correspondences can result in over-estimated FDs, as columns which represent different attributes are merged and create data conflicts, affecting precision. The second reason can further have the effect that two different reference relations are synthesized as a single relation which contains all attributes. As such a relation can only be evaluated for one of the corresponding

reference relations, this also negatively affects recall. The results for earlier stitching stages show that there is an increase in precision to 97.0% for the stitched union tables and to 97.8% for the union tables compared to the final result of this configuration. As each stitching step is based on the schema mapping, the fewer stitching steps are executed, the fewer errors are propagated into the functional dependency discovery and hence cannot reduce the precision of the result.

Normalisation. The final configuration uses the schema matcher, functional dependency discovery, and the normalisation heuristics, i.e., the evaluated relations are not chosen from the discovered FDs for the universal relation, but only from the FDs that hold on the normalised relations. As the normalisation step chooses FDs for decomposition until it can infer a primary key for the normalised relations, it is not dependency preserving and might remove FDs which are actually desirable. During this step, the violation selection heuristics described in Section 9.3.6 are used to decide which FDs are preserved. The results show a small drop in both precision (−2%) and recall (−4.7%) compared to the previous configuration. Based on the evaluation results, it can be concluded that the normalisation step with the proposed heuristics does not have a large negative influence on the results by removing desirable FDs.

The experiments presented in this section show how each component contributes to the SNoW method’s end-to-end performance. It is shown how both context extraction and web table stitching are necessary for the synthesis of high-quality relations, and that this is still possible if errors are introduced by the matching and normalisation components. Based on these results, applications for knowledge base augmentation or table extension can select attributes from the synthesized relations and perform the augmentation using the additional information that is provided by the functional dependencies.

9.5 Corpus Profiling

The SNoW method enables the general-purpose extraction of n-ary relations from large corpora of web tables. Through the application of this method, a data profile of such relations can be created that gives insights about the frequency of relations of higher arity and possible application areas that can benefit from these relations. This section presents such a data profile of n-ary relations in web tables and analyses which types of non-binary relations are found in the Web Data Commons Web Tables Corpus 2015. In order to empirically estimate the answers to these questions, the SNoW method described in Section 9.3 is executed on a corpus of 5.2 million web tables from 86 316 different web sites, the same subset of the WTC 2015 that was used in Chapter 7, using DBpedia as the target knowledge base.

For the creation of this profile, the SNoW method is executed with the normalisation step, which assigns a primary key to every synthesized relation. Hence, this section will refer to primary keys, or just keys, of the relations rather than to functional dependencies as in the previous section. The presented data profile analyses the arity, or size, of these keys and the topical distribution of their attributes. Each of these keys is guaranteed to contain a foreign key that references entities from one of the classes of the target knowledge base, as described in Section 9.3.4 and 9.3.6. This allows for a topical profiling of the synthesized relations similar to the one presented in Chapter 6. The classes represented by the foreign keys will be referred to as the subject class of the synthesized relations, as they are similar to the subject column used to identify the main class of web tables in the previous chapters.

The execution of the SNoW method results in 10 186 synthesized relations with a total of 37 288 non-key attributes which potentially extend the schema of the target knowledge base. This corresponds to 1 340 990 mapped original web tables, which is an increase of 41% compared to the results presented in Chapter 6. With respect to the full corpus, which contains 50.8 million English-language web tables, this corresponds to 2.6% of the web tables that can be interpreted with the DBpedia knowledge base. This percentage is comparable to the results reported in earlier studies [Venetis et al., 2011, Ritze et al., 2016]. Same as in Chapter 6, this limits the scope of the presented data profile, as only web tables which contain data about classes and entities that exist in DBpedia can be analysed. Among the synthesized relations, 37.50% have a key with more than one attribute, i.e., their attributes represent non-binary relations, and 50.47% of these non-binary relations have at least one context attribute in their key.

A manual verification of the synthesized relations with respect to the subject class and candidate key schema mapping to the knowledge base for a random sample of 400 web sites results in a precision of 0.85, weighted by the amount of original web tables. This indicates a high quality of the results and supports the validity of the data profile that is presented in the following.

9.5.1 Topical Profile and Arity of Relations

This section presents statistics about the topical profile of the synthesized relations through the frequency distribution over the high-level classes in the knowledge base. Afterwards, the frequency distribution of synthesized relations of different arities is analysed with respect to their cardinality, i.e., number of tuples, showing that relations of higher arity are only detected for relations with large numbers of tuples.

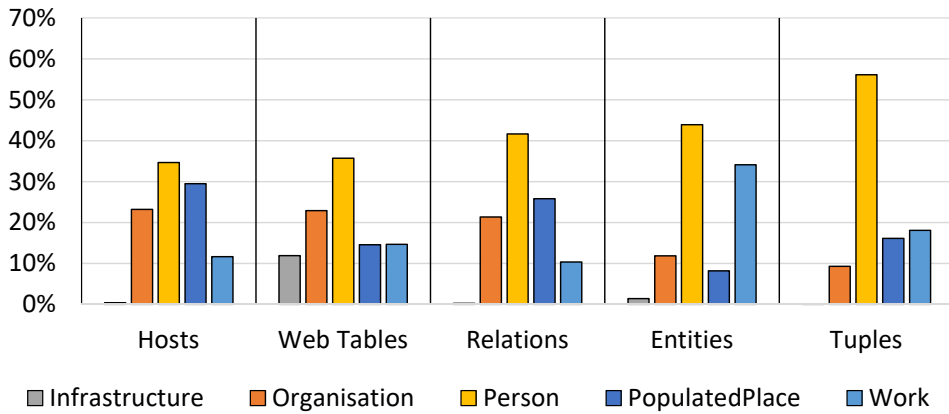


Figure 9.10: Distribution of matched web sites, web tables, relations, tuples, and entities by super class.

Topical Profile

Figure 9.10 shows the frequencies of web sites (hosts) and web tables that were matched to the most frequent high-level classes in the DBpedia type hierarchy. The majority of matches is found for the `Person` class, especially for its `Athlete` sub-class. This is in line with earlier studies and the topical profile presented in Chapter 6, which suggest that the distribution of matching results is related to the distribution of entities in the target knowledge base [Ritze et al., 2016, Hassanzadeh et al., 2015].

The figure further shows the relative frequencies of synthesized relations, tuples and entities that have been extracted from these web tables. Due to the applied normalisation procedure, exactly one relation is synthesized for each class and primary key per web site, and such a relation can contain multiple dependent attributes. The number of synthesized relations hence indicates the diversity of the keys that are required to understand the attributes that are found in the web tables. The amount of extracted entities is the number of rows in the projected tables for each class, as described in Section 9.3.4.

It can be seen that the relative frequencies of the various groups differ from class to class. The `Person` class occurs very frequently among all web sites (35%), web tables (36%), relations (42%), tuples (56%), and entities (44%). In contrast, the `Infrastructure` class occurs only on very few web sites (<1%) and results in only small percentages of the extracted relations, tuples, and entities, but still contributes 12% of all web tables. This indicates a high degree of replication in these web tables. For the `PopulatedPlace` class, a large number of relations (26%) is synthesized for a relatively small percentage of entities (8%), which indicates that most web sites provide information about a rather small set of places in a large number of different contexts, i.e., the places are combined with different other attributes as keys. This results in an average of 21.2 tuples per entity over all synthesized relations, which is the highest for all classes. The opposite can

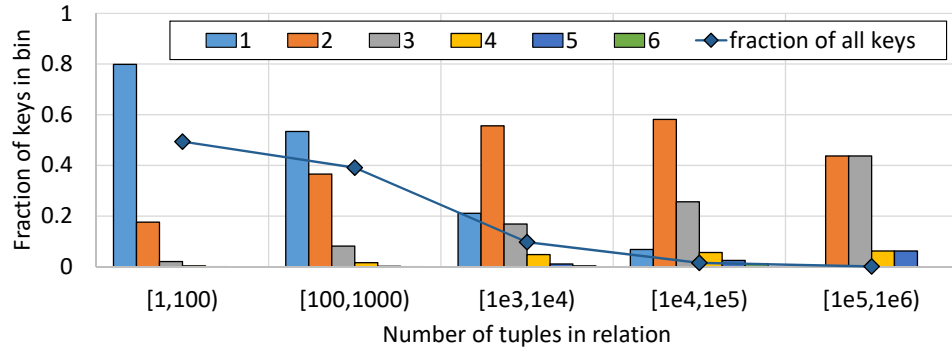


Figure 9.11: Distribution of key sizes by relation size.

be seen for the `Work` class, where rather few relations (10%) are synthesized for a very large percentage of the extracted entities (34%). This indicates that web sites provide information about many different works, but use a small set of different keys, resulting in an average of 5.7 tuples per entity over all synthesized relations.

Key Size Distribution

The following discusses the frequency distribution of keys of different arities in the synthesized relations. Each synthesized relation is assigned a primary key and contains one or multiple dependent attributes, and the arity of the key is the number of attributes in this key. It is further guaranteed that the foreign key that identifies the main entity in each synthesized relation is included in the primary key, which allows for the interpretation of the key sizes: A key of size 1 contains only a reference to an entity and can hence be modelled as binary relation in the knowledge base, and larger keys indicate an n-ary relation.

Figure 9.11 shows the frequency distribution of key sizes by number of tuples. The bars in each group show the fraction of all keys in the group which have the key size indicated by the series, and the line shows the size of each group relative to all synthesized relations. Key sizes larger than 6 are not shown due to their low frequency (the largest discovered key contains 10 attributes). It can be observed that the key size increases with increasing cardinality of the synthesized relations and that large keys cannot be discovered for synthesized relations with few tuples. This suggests that n-ary relations cannot be identified when looking at individual web tables, which have an average size of only 15 rows, and that the stitching of web tables is necessary. The distribution further reveals that synthesized relations with high cardinality rarely are binary relations. This indicates that data providers which publish many web tables with the same schema use these tables to convey more complex information. This means that the largest amounts of data in the analysed web tables are contained in complex, i.e., non-binary relations. This is confirmed by the number of synthesized tuples per key size: Although 62.5% of all synthesized relations are binary (key size 1), they contain only 13.0% of the

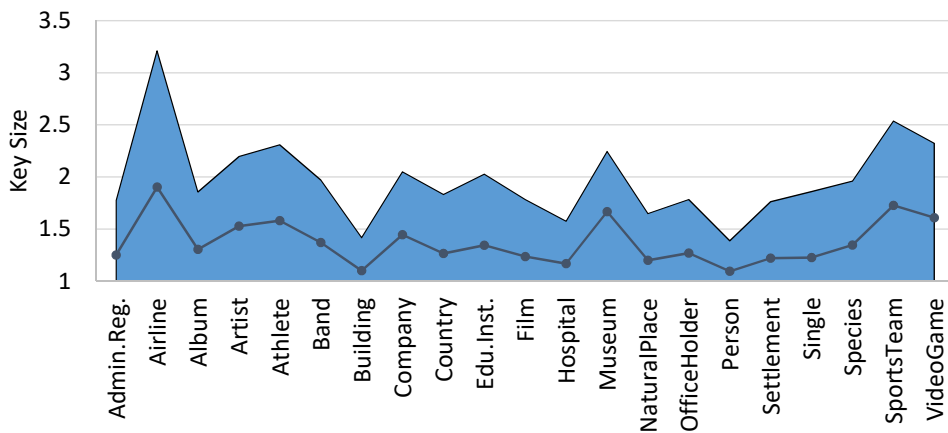


Figure 9.12: Average key size and standard deviation by class.

synthesized tuples. The majority of all extracted data is contained in ternary relations (key size 2), which make up 29.4% of all relations and 56.1% of all tuples, and in quaternary relations (key size 3), which are 6.4% of all relations and contain 19.7% of all tuples.

Figure 9.12 shows the average key size and one standard deviation around the average for the different classes that were encountered. For all classes, keys which contain additional attributes beside the entity identifiers are found. This indicates that n-ary relations are not limited to certain topics in web tables, but rather are a common way of modelling the data.

The figure further shows that the largest keys on average are found for the *Airline* class. Inspection of these synthesized relations reveals that they are actually about air-traffic routes, which are not represented by a class in DBpedia. The same finding holds for synthesized relations assigned to the *Museum* class, which detail the exhibition locations of art pieces that are not in DBpedia, and for synthesized relations assigned to the *SportsTeam* class, which often are about the performance of teams in certain sports events. Other than museums, buildings in general are described by small keys, which can be explained by the specific kind of buildings listed in DBpedia: these are noteworthy, i.e., rather famous buildings, so web sites list interesting facts about them such as building year or height, and their names are mostly unambiguous, so no additional key attributes are required.

9.5.2 Key Attribute Profiling

This section presents a data profile of the contents of the attributes in the keys of the synthesized relations. This provides additional insights about the semantics of these keys and hints at potential use cases for the synthesized relations.

To understand the content of the additional key attributes, a duplicate-based schema matcher is used to create correspondences between the additional attributes in keys of size > 1 and the top-level classes in the DBpedia type hierarchy. This

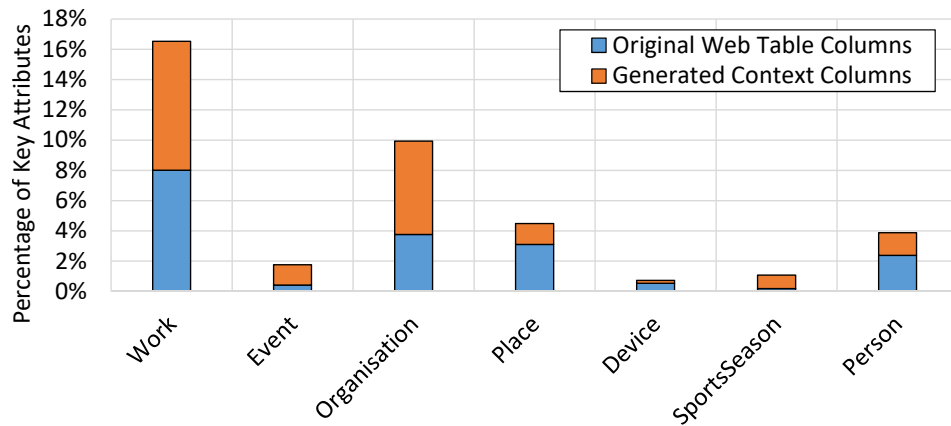


Figure 9.13: Distribution of classes in keys by column type.

matching results in class annotations for 40% of all additional key attributes. Figure 9.13 shows the percentage of key attributes that were matched to the different classes and differentiates between original web table columns and the context columns that were created by the SNoW method. Most of the discovered correspondences are found for the classes *Work*, *Organisation*, and *Person*. The additional key attributes are dominated by correspondences to the class *Work*, while the most frequent subject class as detected during the application of the SNoW method is the class *Person* (compare Figure 9.10). This shift away from the *Person* class can be explained intuitively, as the majority of the synthesized relations already refers to a person through their foreign key, and it would not be expected to find a very large amount of web tables that contain data about the relationship to a second person. The figure further shows that works, events and organisations are discovered more frequently in the generated context columns than in the original web tables, while places, people and devices are more frequently found in the original columns.

Figure 9.14 shows the frequency of class combinations. The groups indicate the subject class assigned to the synthesized relations by the SNoW method, while the bars show the percentage of additional key attributes that were matched to the class indicated by the series. Frequent combinations include *Person* and *Work*, for example musical artists and their songs or albums, writers and their publications or actors and the movies in which they acted; combinations of multiple *Work* entities, such as songs and music albums; as well as multiple *Organisation* entities, for example in web tables describing the results of a match between two sports teams.

Figure 9.15 shows the distribution of correspondences discovered for context attributes by type of context. Most correspondences are found for the attributes created from the page title, which usually contain long strings with multiple entity mentions. Matches for context attributes created from URLs are much less frequent, which is probably due to the fact that URLs contain abbreviated entity names or do not mention any entities at all. The disambiguation attributes that are

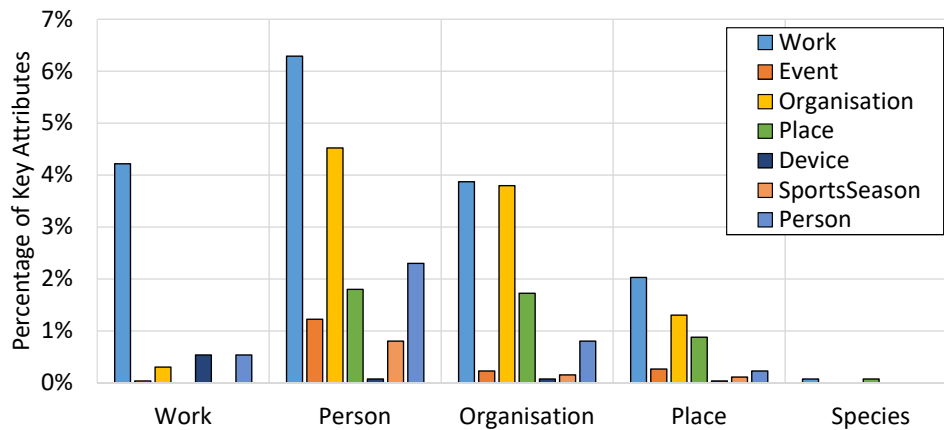


Figure 9.14: Distribution of classes in keys by subject class.

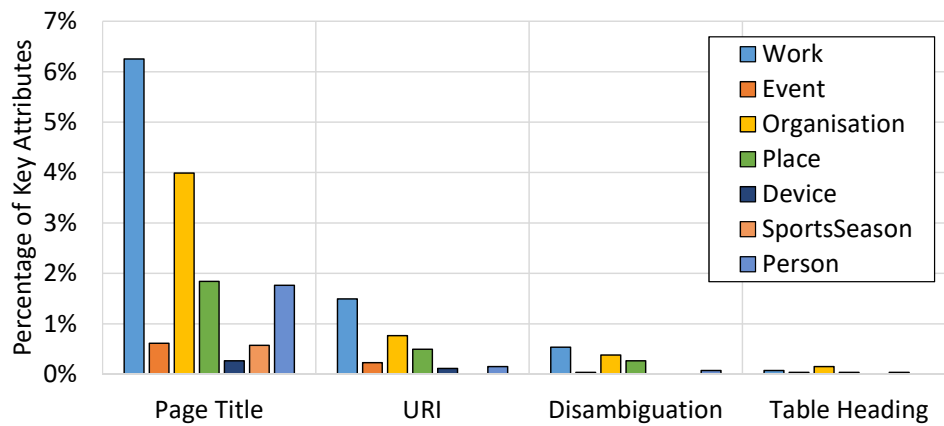


Figure 9.15: Distribution of classes in context attributes by type.

created for values in brackets also often contain values which cannot be linked to the knowledge base, such as “*live version*” for songs.

In addition to linking the additional key attributes to classes in DBpedia, the Stanford NLP SUTime temporal tagger [Chang and Manning, 2012] is used to detect date and time mentions in the additional key attributes. Figure 9.16 shows the distribution of detected date and time values by subject class for different types of columns. This gives more information than an analysis of the detected data types, as especially the page title attributes contain mostly text and are assigned the data type `string`, even if they contain a date or time mention. The result shows that the subject classes `Person` and `Work` occur most frequently together with date and time mentions. It can further be seen that these mentions are primarily found in the context attributes created from the page title. Examples are results of sports events for athletes, the sales figures of works, such as films or video games, or statistics about population or employment for places at a specific point in time.

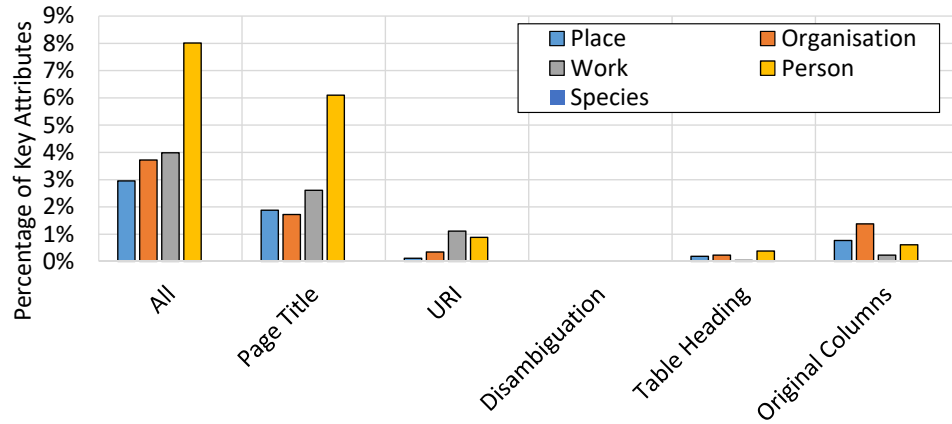


Figure 9.16: Distribution of date and time values in additional key attributes.

Summing up, the presented data profile shows that the additional key attributes follow a different class distribution than the subject classes of the synthesized relations, where the focus is shifted from the `Person` class to the classes `Work` and `Organisation`. The classes `Work`, `Event`, and `Organisation`, as well as date and time mentions, are more often found in context attributes than in original web table attributes. Analysing the different types of generated context attributes reveals that the page title is most informative, as the largest number of entities and dates can be found there. This indicates that further parsing of the page title for attribute extraction is promising for future work. With respect to potential applications, the preceding analysis shows that data sources for web tables can and need to be selected with respect to the specific requirements of the user. A simple example could be to present population numbers for countries over time by selecting web tables which contain this attribute and a date-valued attribute as additional key attribute. However, this also means that sources need to be selected more carefully, as for example employment statistics for different profession are not useful to a user who wants to learn about the total employment in different areas.

9.5.3 Cross-Site Schema Matching.

The previous sections analysed the distributions of topics and the frequencies of different key sizes for the data provided by each web site individually. While this is essential to understand the various types of data that are published through the web tables by different data providers, it gives only limited insights into the semantics of the data in the web table corpus as a whole. Such an understanding can be gained by matching the synthesized relations among each other and analysing their differences.

In the following, the differences in the keys that are found for semantically equivalent attributes on different web sites are analysed. For this analysis, clusters of non-key attributes are created based on their column header and subject class

Table 9.5: Examples of frequent attributes, their average key size and standard deviation, and key attributes in addition to the entity reference. n : number of web sites, μ : average key size, σ : key size standard deviation.

Class	Header	n	μ	σ	Additional Key Attributes	Keys with size		
						1	2	3
Athlete	laps	56	2.11	1.02	event, date	17	24	8
Athlete	position	241	1.17	0.40	event, team	203	37	0
Athlete	team	459	1.41	0.61	event, date	299	139	16
Company	ticker	17	1.18	0.39	-	17	0	0
Country	gold	32	1.47	0.62	event, total	19	11	2
Country	population	76	1.23	0.42	date, group	64	12	0
Country	rank	157	1.34	0.62	event, score	115	32	9
Film	director	13	1.00	0.00	-	13	0	0
Office-Holder	party	92	1.12	0.33	date, term, election	81	11	0
Single	chart	54	1.18	0.47	date, chart	47	5	2
	position							
Sports-Team	pts	35	1.80	0.90	league, games played	16	12	5
Video-Game	sale price	20	1.70	0.57	platform, reg. price	7	12	1

assignment. Table 9.5 shows a selection of the most frequent attributes for different subject classes and their average key size and standard deviation. The column “*Additional Key Attributes*” shows key attributes that were identified in addition to a reference to an entity of the assigned subject class. Many of these key attributes are context attributes, which always have the same header, for example “*page title*”, so the column headers cannot be used for matching. To identify semantically equivalent key attributes, the duplicate-based schema matcher (see Section 9.3.3) is run again on the synthesized relations, followed by a manual inspection of the column contents of the resulting clusters. The attribute names stated in Table 9.5 are the result of this manual inspection. The last three columns in the table show the frequency of keys with the size indicated in the column header. Keys of size 1 always contain an attribute that references the entity of the respective subject class.

The results of this matching show that additional key attributes often contain a reference to a point in time, either explicitly (“*date*”) or implicitly by referring to an event (“*event*”). Such events are for example sport events such as “*2010 FIFA World Cup*” or “*Red Bull Indianapolis Grand Prix 2010*”. It can further be seen that seemingly semantically equivalent attributes, such as “*population*”, are stated with respect to different contexts by the different data providers. This means that

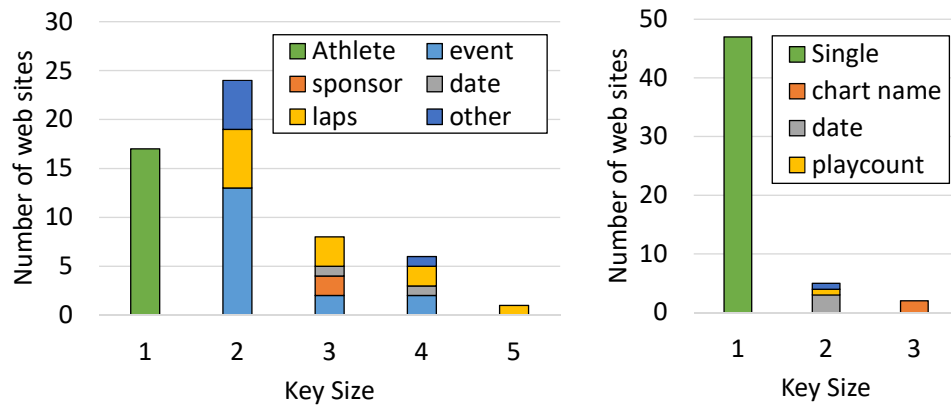


Figure 9.17: Left: Distribution of different groups of keys for the *laps* attribute of athletes. The series “*laps*” indicates relations where the attribute itself is part of the key. Right: Distribution for the *chart position* attribute of singles.

the values that are stated in the respective web tables are not directly comparable and the context, represented by the additional key attributes, must be considered by applications that use these data. It is also important to note that this is not limited to attributes which do not exist in knowledge base, but also for such which have existing properties that are modelled as binary relations in DBpedia.

Key Clusters. Figure 9.17 shows a histogram of different clusters of keys that were found for two exemplary attributes. The left histogram shows the “*laps*” attribute of athletes, which states the number of laps that an athlete completed in a racing event, and the right histogram shows the “*chart position*” attribute of singles. The two most frequent additional key attributes for “*laps*” are “*event*” and “*laps*”. While “*event*” is rather expected, “*laps*” must be clarified: these are relations where the attribute itself is part of the key and determines another attribute such as the current or finishing position. For the “*chart position*” of singles, “*date*” is found as additional key attribute for a key size of two, and “*chart name*” for a key size of three. This seems reasonable, as for web sites which only provide one type of music chart, the position of a single changes depending on the date, while the name of the chart is required if a web site provides data for different charts, such as “*German Top 20*” and “*UK Top 10*”.

Types of N-ary Relations

Among the inspected examples, three different types of relations can be identified: *time-varying*, *composite* and *disambiguation*. These relation types have different characteristics with respect to the integration with the knowledge base, which are discussed in the following.

Time-varying. Time-varying relations describe properties of entities that can change over time, such as the team or position of an athlete.² Such relations often contain a temporal reference in their key attributes. Event mentions in their key attributes can be interpreted as a proxy for a temporal reference, i.e., an athlete was playing for a certain team during the event. In Table 9.5, “*position*” and “*team*” for athletes, “*population*” for countries and “*party*” for office holders fall into this category. The data from time-varying relations can be used by time-aware data fusion methods [Oulabi et al., 2016] or can be directly inserted into knowledge bases which support a time dimension. If a binary relation was assumed, as in most state-of-the-art methods, the temporal context would be lost and either a list of values or an arbitrary single value must be chosen.

Composite. Composite relations describe an entity that is composed of multiple other entities, and the subject class that the matcher recognised represents one of these entities. For example, the “*gold*” attribute for the `Country` class is part of a composite relation “*participation in sports event*”, for which also the attributes “*silver*”, “*bronze*” and “*total*” are found. In Table 9.5, “*laps*” for athletes, “*gold*” and “*rank*” for countries, and “*points*” for sports teams fall into this category. Other than in the case of time-varying attributes, the attributes in composite relations are meaningless for only one of the participating classes, i.e., “*gold*” is neither a meaningful property of a country nor a sports event. If the composite relation is mapped to the class of only one of its elements, it can be assumed that the knowledge base does not contain a corresponding class. The data from composite relations can hence be integrated with the knowledge base by creating a new class. In the case of assuming a binary relation, a nonsensical attribute would be created that cannot be interpreted, as at least one of the participating entities is missing.

Disambiguation. Disambiguations occur in relations which describe entities that are more fine-grained than the entities in the target knowledge base. For example, each entity of the `VideoGame` class in DBpedia can have multiple values for the “*platform*” property (such as “*PC*” or “*XBox*”), i.e., there exists only one entity for all different platform versions of a video game. However, some web sites provide different attribute values, for example for “*sale price*” or “*release date*”, depending on the platform. Such data can either be integrated by adding a new class to the knowledge base (as for composite relations) or, alternatively, the knowledge base can be re-modelled by adding the additional key attribute(s) to the class’ key definition and creating new entities for each of the additional attribute values. If a binary relation was assumed, the attributes would be multi-valued and it is impossible to reconstruct which combinations of their values are actually valid.

²Position is limited to string-valued attributes and refers to the athlete’s position in the field of play, such as “*centre forward*” or “*goalkeeper*”.

The preceding analysis of keys for semantically equal attributes has revealed that information extraction that is limited to binary relations cannot capture the complexity of the information that is conveyed in web tables. It has been shown that the same attribute can be presented with different keys, which means that the values provided in the respective web tables are not comparable without considering their context. This could explain the low quality of facts extracted from web tables, which was observed in several earlier studies that assumed binary relations [Ritze et al., 2016, Dong et al., 2014a]. If a method extracts values for an attribute from different web tables without considering that their context is a different point in time, a different event or that the entities are described on a different level of granularity, the quality of the data extracted from web tables can be under-estimated. Further, different types of n-ary relations were identified and described with respect to their integration with a knowledge base. This shows that not only the semantic annotation of n-ary relations in web tables, but also their possible representation in knowledge bases needs more attention by the research community.

9.6 Conclusion

This chapter presented SNoW, the first method to synthesize n-ary relations from web tables for the purpose of knowledge base extension. First, related work was introduced in Section 9.2. Then, Section 9.3 described the proposed method along the different steps of the overall workflow. Section 9.4 presented an experimental evaluation of the method and described the datasets that are used for the evaluation. Finally, Section 9.5 presented the results of using the proposed method to profile a corpus of 5 million web tables.

This chapter made the following contributions:

- **Method:** This chapter presented the first method to synthesize general n-ary relations from web tables. This is achieved by the extraction of additional context data from the web page containing the web tables, and stitching multiple web tables from the same web site, which enables functional dependency discovery on the stitched tables. The experimental evaluation showed that the proposed method achieves an average F1-measure of 0.93 for the schema matching task and 0.85 for the relation synthesis, which is an improvement of 0.17 over a perfect extractor for binary relations.
- **Data Profile:** This chapter further presented the first data profile that not only considers binary relations in web tables, but also analyses n-ary relations in a corpus of 5 million web tables. The profile shows that web tables contain large amounts of n-ary relations which are quite diversely modelled. The data profile reveals that a large fraction of the data in web tables is of higher complexity than previously assumed, as it is shown that 37.5% of all synthesized relations and 87% of all synthesized tuples have a key that con-

tains more than one attribute. The additional key attributes are generated from the context of the web tables in 50.47% of the cases, showing that only focusing on the content of the web tables without considering the web page that contains them is insufficient in many cases.

- **Evaluation Dataset:** Along with the evaluation of the presented method, large evaluation datasets were created and published for the task of identifying n -ary relations in web tables. These datasets consist of more than 300 thousand web tables. By publishing the datasets, the replicability of the results is ensured and a first foundation for comparing methods that synthesize n -ary relations from web tables is provided.

The method presented in this chapter is the first that approaches the problem of extracting general n -ary relations from web tables. Earlier methods were limited to specific types of n -ary relations, such as ternary relations that include a time dimension. Most of the work in the area of web tables, however, assumes that binary relations are sufficient to understand their content. The profiling that was presented in this chapter demonstrated that such an assumption is not justified, as a large fraction of the web tables, which can be interpreted with the DBpedia knowledge base, represents n -ary relations. Ignoring this fact leads to incorrectly interpreted data and may be a reason for the low quality of previous methods that extract statements from web tables for knowledge base augmentation.

For a correct extraction and interpretation of the data in web tables, this chapter has shown that it is necessary to consider additional values which are stated on the web page that contains the web table, and that multiple web tables need to be combined such that the functional dependencies among their attributes can be discovered. These two changes are essential for the high quality that is achieved by the SNoW method, which is the first to apply this specific combination of methods to solve the task of synthesizing relations from web tables.

The data profile that was generated through the application of this method revealed that the data in web tables is of much higher complexity than previously assumed, and shows a clear need of considering this complexity in applications that use web tables. Only if the data is correctly interpreted can an application provide useful results to a user. However, the profile also shows new possibilities for applications, as the data from web tables is often more detailed than the knowledge contained in current knowledge bases. This means, instead of just looking up a certain value as is possible with a knowledge base, a user can be enabled to drill-down into detailed statements based on various dimensions, for example by exploring the average annual wage not only by state, but also by profession and time.

Chapter 10

Conclusion

This chapter summarises the contributions of this thesis, outlines its research impact, and discusses limitations as well as directions for future work. The range of topics covered in this thesis spans the complete process required to use web tables for knowledge base augmentation. From extracting raw web tables from web pages, to interpreting the semantics of the extracted web tables, profiling the contents of the created corpora for slot filling, and finally generating relevant and complete candidates for schema extension.

10.1 Summary

This section summarises the content of this thesis and highlights the individual findings and contributions. It is organised in the same way as the overall thesis and discusses its three parts in order.

Public Web Table Corpora

The data profiles for web tables that are presented throughout this thesis are based on two large-scale web table corpora which were created during the work on this thesis. The publication of these corpora and their data profiles changed the field of web table research by making it more accessible, as earlier corpora were privately owned and could not be analysed by other researchers. Through the publication of the Web Data Commons Web Tables Corpora (WTC), which represent two of only three publicly available, large-scale corpora of web tables, researchers can now directly access and analyse the properties of web tables, without going through the involved process of crawling billions of web pages and re-implementing the necessary table detection and structural analysis methods that are required to produce such corpora. This reduces the time spent until valuable contributions can be made and leads to more transparency, as methods and their results based on public corpora are reproducible and verifiable.

The process for the extraction of web tables starts with the identification of tables in HTML pages which actually contain data, so-called content tables, opposed to tables which are used for layout purposes. After the content tables have been identified, their structural analysis spans the detection of header rows, column data types, units, and the subject column of the tables. For all these steps, several heuristics or supervised machine learning approaches have been proposed in the literature. The implementation of such methods as open-source enabled the creation of the Web Data Commons Web Tables Corpus 2012 and 2015, which contain millions of content web tables and have been analysed throughout this thesis.

Comparing the web table corpora that are used in the literature showed that only very limited information is available, as most of the corpora are privately owned and not available to other researchers. This entry boundary to research on large-scale methods for web tables was removed by publishing the two described corpora, which contain 147 million and 233 million content tables, respectively. Along with the corpora, data profiles that describe the contained web tables with respect to their table type, size, origin, and data types were published. These data profiles show that the majority of the web tables is very small, with a median of only six rows in both created corpora, and that between 35% and 52% of all columns are non-textual columns, i.e., contain for example numbers or dates. These are two important characteristics for the semantic interpretation of the web tables, which have barely been addressed by the literature on the topic.

Matching Web Tables to a Knowledge Base

After the extraction and structural analysis of web tables, the next step is to create a semantic interpretation which enables applications to effectively retrieve relevant web tables and use their data for several applications such as query or question answering or knowledge base augmentation. However, the field of semantic table interpretation is lacking agreement on the set of tasks that need to be solved as well as on the evaluation methodology and datasets that are used to measure the quality of the proposed methods. Many approaches only consider web table columns that contain named entities and either annotate individual columns with classes, pairs of columns with properties or link the entities mentioned in the columns to a knowledge base. This limits the comparability of the methods and casts doubts on their practical applicability.

The publication of the open-source T2K Match algorithm, which solves all tasks that are necessary to extract triples for columns of all data types from a web table, and the publication of the T2D gold standard for these tasks create the foundation for more transparency and comparability of methods in this area. The T2K Match algorithm annotates web tables with classes, pairs of columns with properties, and cells with entities from a knowledge base. It further considers columns which do not contain named entities. The algorithm achieves an F1-measure of 0.94 for the class annotation, 0.82 for the entity annotation, and 0.7 for the rela-

tion annotation task. A comparison with other methods for the entity annotation task shows that T2K Match outperforms other approaches that were available at the time of its original publication and still achieves comparable results to approaches that have been published afterwards.

The application of the T2K Match algorithm to the WTC 2012 and the creation of a data profile for this corpus with respect to classes, properties, and entities from the DBpedia knowledge base resulted in the currently most-detailed profile of any web table corpus. This data profile shows that ca. 1 million of the 33 million English-language web tables in the WTC 2012 can be mapped to the DBpedia knowledge base, with more than half a million pairs of columns annotated with properties and almost 14 million cells annotated with entities. The data profile further revealed that the majority of the 7.9 million triples that can be generated from the web tables based on the annotations are not for named entity columns, but rather for numerical (2.8 million) and date (3.4 million) values.

Further analysis showed that the current evaluation datasets for semantic table interpretation are biased towards large web tables and the quality of algorithms in this area is over-estimated. Experiments on a random sample showed that such algorithms cannot handle very small web tables, which constitute the majority of web tables in all corpora for which the size distribution of web tables is known. This problem is addressed by table stitching, which was shown to drastically improve the performance of existing semantic table interpretation methods, especially for small web tables. Table stitching is the process of merging web tables based on their schema into larger tables, which increases the amount of different values that a semantic table interpretation method can exploit. This improves the reliability of column statistics, such as uniqueness, which is important for the detection of the correct subject column, and increases the amount of different values that can be used for similarity calculation. The application of this method improves the F1-measure of T2K Match by 0.38 points for the relation annotation task on a random sample of web tables and improves the precision of the generated triples from the web tables in this sample from 0.39 to 0.59.

Extending the Schema of a Knowledge Base

The fine-grained profile of the WTC 2012 showed that the majority of the columns in the web tables that can otherwise be annotated with classes and entities have no corresponding properties in the DBpedia knowledge base. These columns are hence candidates for new properties and the extension of the knowledge base's schema. The state of the art in schema extension from web tables focusses on methods that rank these candidates according to their relevance for the respective class in the knowledge base, for example by considering their co-occurrence frequencies with the existing properties. However, these approaches ignore that the columns in a web table can be dependent on their context, i.e., other columns or data on the web page outside of the web table, and may hence result in incomplete property candidates which cannot be interpreted individually.

To address this issue, a data profile for attributes which are candidates for schema extension was created that shows that the majority of the candidates (62%) does not comply with the assumption that all columns in a web table can be interpreted based on their own values and the values in the subject column alone. This assumption is derived from the data model used in most semantic table interpretation approaches, which states that every column in a web table is in a binary relation with the subject column.

Based on this finding, the first method for the general extraction of n-ary relations from web tables was proposed, which creates additional explanatory attributes from the context of the web tables. These n-ary relations contain all additional attributes which are required to interpret the values and are hence better candidates for schema extension. Through the application of this method, it was shown that web tables contain information of a much higher complexity than previously assumed. The created data profile shows that 37.5% of the created relations require at least one additional attribute beside the subject column, and for 50.5% of these relations at least one of the additional attributes is not a column of the original web tables, but needs to be extracted from the web pages containing the web tables. The proposed method is the first to consider the problem of the general extraction of n-ary relations from web tables, and shows that this task needs more attention from the research community to enable web table-based applications to produce results of high quality. To support additional research in this area, the used evaluation dataset, which contains more than 300 thousand web tables, as well as the implementation of the method that sets a benchmark for other approaches to compare to, are made publicly available.

10.2 Research Impact

This section discusses the research impact of the methods and results presented in this thesis. Especially the Web Data Commons Web Tables Corpus 2015, the T2D gold standard, and the open-source T2K Match algorithm have gained wide attention and are frequently used for the evaluation of new approaches.

After the publication of the WTC 2012 and the source code used for its extraction, Eberius et al. [Eberius et al., 2015] extended the code and published a similar corpus based on a more recent web crawl. In turn, the WTC 2015 incorporated the additions made by Eberius et al. and further extended the extraction framework. Various parts of the extraction methodology used in the WTC 2015 have further been adopted by other authors, such as the table type taxonomy [Nishida et al., 2017] or the subject column detection heuristic [Saleiro et al., 2017]. This shows how the open publication of web table corpora and the code that is necessary to create them can benefit a large community of researchers and enables independent contributions that improve the quality of the available resources.

In the area of dataset search, Zhu et al. use the WTC 2015 for the evaluation of their search methods that speed up the retrieval of related tables [Zhu et al., 2016, Rao and Zhu, 2016, Zhu et al., 2017, Zhu et al., 2019]. Kacprzak et al. also refer to the WTC 2015 to show the relevance of dataset search methods [Kacprzak et al., 2017] and Miller analyses the WTC 2015 as an example of a data lake [Miller, 2018]. This highlights the value of the created web table corpora for the evaluation of methods for large-scale data processing and shows that the WTC 2015 is already a well-known and established dataset in different research communities.

In the area of semantic table interpretation, Ritze and Bizer use the WTC 2015 to sample web tables to create an extended version of the T2D gold standard and use it to evaluate various groups of features for its different tasks [Ritze and Bizer, 2017]. The T2D gold standard is further used for the evaluation of the methods proposed by Pham et al., Efthymiou et al., and Ermilov and Ngomo [Pham et al., 2016, Ermilov and Ngomo, 2016, Efthymiou et al., 2017]. Methodologically, the T2K Match algorithm presented in Chapter 5 is the basis of T2K Match++ [Ritze, 2017] and the foundation of the data integration methods of the DS4DM Project¹ and the corresponding extension for the RapidMiner² data mining tool [Gentile et al., 2016, Kleppmann et al., 2018]. Oulabi and Bizer further use the WTC 2015 to create a data profile of timestamp metadata [Oulabi and Bizer, 2017] and present a method for the entity-set-completion task using this corpus [Oulabi and Bizer, 2019]. This further corroborates the impact of the work presented in this thesis for the area of semantic table interpretation, which has become widely recognised and is used as a reference for the evaluation of new approaches.

10.3 Limitations & Future Work

This section discusses the limitations of the work presented in this thesis and gives directions for future work in the areas of semantic table interpretation and knowledge base augmentation.

The data profile of the two large-scale web table corpora that was presented in this thesis was created through matching the web tables to the DBpedia knowledge base, i.e., it can only give insights into the topical overlap between the web table corpora and this knowledge base. Although there is no related work on the same level of detail, literature suggest that similar results could be obtained with other knowledge bases, such as YAGO or Wikidata [Hassanzadeh et al., 2015, Dong et al., 2014a]. Concerning the coverage of the analysed web tables, i.e., those that could be matched to the knowledge base, it must be stated that current knowledge bases are only useful to annotate a small percentage of the extracted web tables.

A direction for future work is to try and increase this coverage. For example, the stitching method presented in Chapter 7 resolved a methodological problem that could cause a low coverage and should be considered in all future profiling

¹<http://ds4dm.de/>

²<https://rapidminer.com/>

efforts. Also, methods from the area of Open Information Extraction have shown promising results for the large-scale annotation of web tables. For example, Gupta et al. [Gupta et al., 2014] found that they can increase the coverage of annotated attributes by a factor of 5.7 compared to Freebase when using the attribute names that they extracted from web pages and a search engine query stream. In general, such methods collect entity and attribute names from large, unstructured text corpora, which can then be used to annotate web tables. However, the larger coverage achieved by these methods comes with the drawback of higher uncertainty due to noisy extractions during the open information extraction part and does not immediately explain the semantics of the web tables, as the annotations are text tokens which themselves need to be interpreted. Another possibility is that the web tables which cannot be matched to a knowledge base simply do not contain any entities that could be linked, and can only be understood in the context of the web page that contains them. For example for product offers on the web, it is common to place entity tables, i.e., web tables which only describe a single entity (see Section 4.3.1), on the web pages with product offers to describe the technical details of the respective product [Qiu et al., 2015].

With the increased quality of semantic table interpretation methods and the high complexity of the relations that can be synthesized using the method presented in Chapter 9, there is further a need to discuss how this kind of information should be stored. Current knowledge bases do not necessarily support the storage of different levels of granularity or complex relations such as those found in the web tables. While the schema of DBpedia provides limited support for few specific n -ary relations, such as the results of Olympic games or snooker competitions, Wikidata enables the general annotation of statements with additional properties and might be better suited for the augmentation with the complex data from web tables.

In this context, a question that also needs to be addressed is that of the level of detail of information that is desirable for the augmentation of general-purpose knowledge bases, such as DBpedia or Wikidata. For example, in-depth statistics of an athlete's performance in a specific match might not be of general interest and would not increase the value of these knowledge bases. A different direction for future work could hence be the construction of additional, highly detailed knowledge bases from web tables. Such an approach could extend the method presented in Chapter 9 by integrating the synthesized relations across the different web sites in the web table corpus and organise the discovered attributes in different hierarchies according to their context, similar to the approach proposed by Halevy et al. [Halevy et al., 2016] that structures complex attributes based on their names.

List of Figures

1.1	A web table listing airlines and their attributes.	2
1.2	A web table showing employment and wage statistics for different states and a specific profession, which is stated outside of the web table.	3
2.1	Classification of data profiling tasks (reproduced from [Naumann, 2014]).	16
2.2	The Data Integration Process	19
2.3	Matcher components according to the classification systems of Gal and Sagi, and Lee et al.	24
3.1	An example of an RDF graph.	33
3.2	Distribution of DBpedia entities by class.	38
3.3	Density of selected DBpedia properties.	39
4.1	Overview of published research about web tables.	50
4.2	Table Type Taxonomy	52
4.3	Examples of different table types.	53
4.4	Comparison of the distributions of web tables by rows and columns over different corpora.	63
4.5	WTC 2012: Distribution of web tables over rows and columns. . .	67
4.6	WTC 2012: Distribution of web table columns over data types. . .	67
4.7	WTC 2012: Distribution of web tables over top-level domains. . .	67
4.8	WTC 2015: Table Metadata.	71
4.9	WDC Web Tables Corpus 2015: Distribution of tables over table types.	72
4.10	WDC Web Tables Corpus 2015: Distribution of web tables by rows and columns.	72
4.11	WDC Web Tables Corpus 2015: Distribution of web tables by top-level domain.	72
5.1	The Web Table Data Model.	82
5.2	T2D: Data Type Distribution.	95
5.3	T2D: Table Size Distributions.	96

5.4	T2D: Property Frequency Distribution.	97
5.5	T2D: Class distribution for instance-level gold standard.	98
5.6	T2D: Average number of rows and columns and their annotations with entities and properties by class.	98
5.7	T2D: User interface of the annotation tool.	100
5.8	T2D: User interface for table-level metadata.	100
5.9	T2D: User interface for column-level metadata.	101
5.10	T2D: User interface for row-level metadata.	101
5.11	T2K Match Workflow.	103
5.12	Running Example: Example Table.	104
5.13	Running Example: Generated Candidate Entities.	106
5.14	Running Example: Calculated Similarity Values.	107
5.15	Running Example: Voting for Schema Correspondences.	109
5.16	Running Example: Iterative Update.	110
5.17	F1-measure for different high-level classes.	113
5.18	F1-measure for different table sizes.	113
6.1	Distribution of DBpedia entities and correspondences for matched web tables by class.	129
6.2	Distribution of DBpedia properties and correspondences for matched web tables by class.	129
6.3	Distribution of correspondences by number of web tables.	131
6.4	Distribution of group sizes.	131
6.5	Distribution of relevant groups by size.	134
7.1	Example how information about video games is broken up into multiple small web tables on different pages.	146
7.2	Number of schemata per web site.	153
7.3	Number of web tables per schema.	154
7.4	Example of tables that are missed by the union approach.	155
7.5	Graph-based refinement	158
7.6	Precision and recall for all matcher configurations.	163
7.7	Evaluation of extracted values.	169
7.8	Tables per web site at different stitching steps.	170
7.9	Influence of table characteristics on matching performance.	171
8.1	T2D: Evaluation of matching methods for unknown attributes.	189
8.2	T2D: Precision@k and recall@k achieved by the different ranking methods using the duplicate-based matcher.	190
8.3	T2D: Rank of the first cluster matching the removed attribute.	191
8.4	WDC 2012: Number of candidate columns and clusters for schema extension.	192
8.5	WDC 2012: Precision@15 for the attribute ranking experiment.	193
8.6	WDC 2012: Rank of the first cluster matching the removed attribute.	194

8.7	Category Hierarchy	198
8.8	Example tables with column category annotation.	199
9.1	Introduction of the SNoW method.	213
9.2	Example web table.	216
9.3	Overview of the SNoW method.	217
9.4	Example for context extraction from the web page surrounding a web table.	218
9.5	Holistic Global Matching.	221
9.6	Example of projection of known classes.	223
9.7	Example of two tables with the same schema that satisfy different sets of functional dependencies.	227
9.8	Example of the normalisation process.	228
9.9	Results of the relation synthesis experiments.	236
9.10	Distribution of matched web sites, web tables, relations, tuples, and entities by super class.	241
9.11	Distribution of key sizes by relation size.	242
9.12	Average key size and standard deviation by class.	243
9.13	Distribution of classes in keys by column type.	244
9.14	Distribution of classes in keys by subject class.	245
9.15	Distribution of classes in context attributes by type.	245
9.16	Distribution of date and time values in additional key attributes.	246
9.17	Key Clusters.	248

List of Tables

3.1	Overview of common knowledge bases (reproduced from [Paulheim, 2017]).	35
4.1	Overview of web table corpora (K=thousand, M=million, B=billion).	63
5.1	T2K Match Evaluation.	112
5.2	Example of a web table that is hard to match due to very similar values in two columns with different semantics.	114
5.3	Example of a web table that is hard to match due to ambiguous entity names and no additional properties that exist in the knowledge base.	114
5.4	Comparison of different methods for the entity annotation task.	116
6.1	Characteristics of the analysed web table corpus.	124
6.2	Analysed Corpus: Most frequent web sites and column headers.	125
6.3	Aggregated correspondence statistics. Each row represents the aggregated statistics for the class mentioned in the first column and all its subclasses.	127
6.4	Distribution of values by data type.	131
6.5	Most frequent classes, properties and entities.	132
6.6	Largest groups and evaluation according to the Local-Closed-World Assumption.	135
6.7	Number of existing and new triples and evaluation results by fusion strategy.	138
6.8	Data fusion performance by data type.	139
6.9	Data fusion performance by class.	140
6.10	Examples of frequent properties after data fusion.	142
7.1	Datasets for schema matching of union tables.	160
7.2	Comparison of all schema matchers (F1-measure, best configuration for each dataset is shown in boldface).	161
7.3	Datasets for stitched union table to knowledge base matching.	164
7.4	Evaluation of T2K Match for web tables, union and stitched union tables.	165

7.5	Results of running T2K Match and COMA with web tables, union tables and stitched union tables.	166
8.1	Overview of tasks and evaluation criteria used in related work for schema extension from web tables.	183
8.2	Highest ranked attributes for class <code>Drug</code>	195
8.3	Highest ranked attributes for class <code>Airline</code>	196
8.4	Distribution of column categories in the manually annotated sample.	201
8.5	Features used for column classification.	205
8.6	Confusion matrix for the column category classifier.	205
8.7	Distribution of column categories in the WDC Web Tables Corpus 2012.	206
8.8	Frequent attributes that are classified as binary relations.	207
9.1	An example of a table with null values.	225
9.2	Annotated Dataset Statistics.	232
9.3	Examples of FDs in the annotated datasets.	233
9.4	Experimental results for schema matching and relation synthesis (F1-measure).	235
9.5	Examples of frequent attributes.	247

Bibliography

- [Akbik and Löser, 2012] Akbik, A. and Löser, A. (2012). Kraken: N-ary facts in open information extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 52–56. Association for Computational Linguistics.
- [Arasu and Garcia-Molina, 2003] Arasu, A. and Garcia-Molina, H. (2003). Extracting structured data from web pages. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348. ACM.
- [Atzeni and Morfuni, 1986] Atzeni, P. and Morfuni, N. M. (1986). Functional dependencies and constraints on null values in database relations. *Information and Control*, 70(1):1–31.
- [Augenstein et al., 2012] Augenstein, I., Padó, S., and Rudolph, S. (2012). Lodi-fier: Generating linked data from unstructured text. In *Extended Semantic Web Conference*, pages 210–224. Springer.
- [Aumuellner et al., 2005] Aumuellner, D., Do, H.-H., Massmann, S., and Rahm, E. (2005). Schema and ontology matching with coma++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908. ACM.
- [Balakrishnan et al., 2015] Balakrishnan, S., Halevy, A. Y., Harb, B., and et al. (2015). Applying webtables in practice. In *Conference on Innovative Data Systems Research CIDR*.
- [Bauckmann et al., 2007] Bauckmann, J., Leser, U., Naumann, F., and Tietz, V. (2007). Efficiently detecting inclusion dependencies. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1448–1450. IEEE.
- [Bernstein, 1976] Bernstein, P. A. (1976). Synthesizing third normal form relations from functional dependencies. *ACM Transactions on Database Systems (TODS)*, 1(4):277–298.
- [Bernstein et al., 2011] Bernstein, P. A., Madhavan, J., and Rahm, E. (2011). Generic schema matching, ten years later. *Proceedings of the VLDB Endowment*, 4(11):695–701.

- [Bhagavatula et al., 2015] Bhagavatula, C. S., Noraset, T., and Downey, D. (2015). TabEL: entity linking in web tables. In *International Semantic Web Conference*, pages 425–441. Springer.
- [Bilenko et al., 2003] Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., and Fienberg, S. (2003). Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23.
- [Bilke and Naumann, 2005] Bilke, A. and Naumann, F. (2005). Schema matching using duplicates. In *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, pages 69–80. IEEE.
- [Bleiholder and Naumann, 2009] Bleiholder, J. and Naumann, F. (2009). Data fusion. *ACM Comput. Surv.*, 41(1):1–41.
- [Bollacker et al., 2007] Bollacker, K., Tufts, P., Pierce, T., and Cook, R. (2007). A platform for scalable, collaborative, structured information integration. In *Intl. Workshop on Information Integration on the Web (IIWeb'07)*, pages 22–27.
- [Braunschweig et al., 2015] Braunschweig, K., Thiele, M., Eberius, J., and Lehner, W. (2015). Column-specific Context Extraction for Web Tables. In *Proc. of the 30th Annual ACM Symposium on Applied Computing, SAC '15*, pages 1072–1077.
- [Brickley et al., 2014] Brickley, D., Guha, R. V., and McBride, B. (2014). Rdf schema 1.1. *W3C recommendation*, 25:2004–2014.
- [Broder et al., 2000] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000). Graph structure in the web. *Computer networks*, 33(1-6):309–320.
- [Broder et al., 1997] Broder, A. Z., Glassman, S. C., Manasse, M. S., and Zweig, G. (1997). Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8):1157–1166.
- [Bronzi et al., 2013] Bronzi, M., Crescenzi, V., Merialdo, P., and Papotti, P. (2013). Extraction and integration of partially overlapping web sources. *Proceedings of the VLDB Endowment*, 6(10):805–816.
- [Bryl et al., 2015] Bryl, V., Bizer, C., and Paulheim, H. (2015). Gathering alternative surface forms for dbpedia entities. In *NLP-DBPEDIA@ ISWC*, pages 13–24.
- [Buche et al., 2013] Buche, P., Dibie-Barthelemy, J., Ibanescu, L., and Soler, L. (2013). Fuzzy web data tables integration guided by an ontological and terminological resource. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):805–819.

- [Busse et al., 1999] Busse, S., Kutsche, R.-D., Leser, U., and Weber, H. (1999). Federated information systems: Concepts, terminology and architectures. *Forschungsberichte des Fachbereichs Informatik*, 99(9):1–38.
- [Cafarella et al., 2008a] Cafarella, M., Halevy, Alonand Zhang, Y., Wang, D. Z., and Wu, E. (2008a). Uncovering the Relational Web. In *Proc. of the WebDB Workshop*.
- [Cafarella et al., 2009] Cafarella, M. J., Halevy, A., and Khoussainova, N. (2009). Data Integration for the Relational Web. *Proc. of the VLDB Endow.*, 2:1090–1101.
- [Cafarella et al., 2008b] Cafarella, M. J., Halevy, A., Wang, D. Z., Wu, E., and Zhang, Y. (2008b). WebTables: Exploring the Power of Tables on the Web. *Proc. of the VLDB Endow.*, 1:538–549.
- [Cannavicchio et al., 2018] Cannavicchio, M., Barbosa, D., and Merialdo, P. (2018). Towards annotating relational data on the web with language models. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1307–1316. International World Wide Web Conferences Steering Committee.
- [Carlson et al., 2010] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3. Atlanta.
- [Carmel et al., 2001] Carmel, D., Cohen, D., Fagin, R., Farchi, E., Herscovici, M., Maarek, Y. S., and Soffer, A. (2001). Static index pruning for information retrieval systems. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50. ACM.
- [Chang and Manning, 2012] Chang, A. X. and Manning, C. D. (2012). Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 2012, pages 3735–3740.
- [Chaudhuri et al., 2003] Chaudhuri, S., Ganjam, K., Ganti, V., and Motwani, R. (2003). Robust and Efficient Fuzzy Match for Online Data Cleaning. In *Proc. of the 2003 ACM SIGMOD Int. Conference on Management of Data*, pages 313–324, New York, NY, USA. ACM.
- [Chen et al., 2000] Chen, H.-H., Tsai, S.-C., and Tsai, J.-H. (2000). Mining tables from large scale html texts. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 166–172. Association for Computational Linguistics.
- [Chu et al., 2015] Chu, X., Morcos, J., Ilyas, I. F., Ouzzani, M., Papotti, P., Tang, N., and Ye, Y. (2015). Katara: A data cleaning system powered by knowledge

- bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1247–1261. ACM.
- [Church and Hanks, 1990] Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.
- [Cochinwala et al., 2001] Cochinwala, M., Kurien, V., Lalk, G., and Shasha, D. (2001). Efficient data reconciliation. *Information Sciences*, 137(1-4):1–15.
- [Codd, 1970] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.
- [Codd, 1972] Codd, E. F. (1972). Further normalization of the data base relational model. *Data base systems*, pages 33–64.
- [Cohen, 2000] Cohen, W. W. (2000). Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems (TOIS)*, 18(3):288–321.
- [Cohen et al., 2003] Cohen, W. W., Ravikumar, P., Fienberg, S. E., et al. (2003). A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 2003, pages 73–78.
- [Collins and Duffy, 2002] Collins, M. and Duffy, N. (2002). Convolution kernels for natural language. In *Advances in neural information processing systems*, pages 625–632.
- [Craswell, 2009] Craswell, N. (2009). Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer.
- [Crescenzi et al., 2001] Crescenzi, V., Mecca, G., Merialdo, P., et al. (2001). Roadrunner: Towards automatic data extraction from large web sites. In *VLDB*, volume 1, pages 109–118.
- [Crestan and Pantel, 2010] Crestan, E. and Pantel, P. (2010). A fine-grained taxonomy of tables on the web. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1405–1408. ACM.
- [Crestan and Pantel, 2011] Crestan, E. and Pantel, P. (2011). Web-scale Table Census and Classification. In *Proc. of the 4th Int. Conf. on Web Search and Data Mining (WSDM)*, pages 545–554. ACM.
- [Das Sarma et al., 2012] Das Sarma, A., Fang, L., Gupta, N., Halevy, A., Lee, H., Wu, F., Xin, R., and Yu, C. (2012). Finding Related Tables. In *Proc. of the Int. Conference on Management of Data*, pages 817–828.
- [Date, 2012] Date, C. (2012). *Database design and relational theory: normal forms and all that jazz*. O’Reilly Media, Inc.

- [De Marchi et al., 2002] De Marchi, F., Lopes, S., and Petit, J.-M. (2002). Efficient algorithms for mining inclusion dependencies. In *International Conference on Extending Database Technology*, pages 464–476. Springer.
- [Do and Rahm, 2002] Do, H.-H. and Rahm, E. (2002). Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 610–621. VLDB Endowment.
- [Doan et al., 2012] Doan, A., Halevy, A., and Ives, Z. (2012). *Principles of data integration*. Elsevier.
- [Dong et al., 2014a] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmman, T., Sun, S., and Zhang, W. (2014a). Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *Proc. of the 20th SIGKDD*, pages 601–610.
- [Dong et al., 2014b] Dong, X. L., Gabrilovich, E., Heitz, G., Horn, W., Murphy, K., Sun, S., and Zhang, W. (2014b). From data fusion to knowledge fusion. *Proceedings of the VLDB Endowment*, 7(10):881–892.
- [Dong et al., 2015] Dong, X. L., Gabrilovich, E., Murphy, K., Dang, V., Horn, W., Lugaresi, C., Sun, S., and Zhang, W. (2015). Knowledge-based Trust: Estimating the Trustworthiness of Web Sources. *Proc. of the VLDB Endow.*, 8(9):938–949.
- [Draisbach and Naumann, 2009] Draisbach, U. and Naumann, F. (2009). A comparison and generalization of blocking and windowing algorithms for duplicate detection. In *Proceedings of the International Workshop on Quality in Databases (QDB)*, pages 51–56.
- [Eberius et al., 2015] Eberius, J., Braunschweig, K., Hentsch, M., Thiele, M., Ahmadov, A., and Lehner, W. (2015). Building the dresden web table corpus: A classification approach. In *Big Data Computing (BDC), 2015 IEEE/ACM 2nd International Symposium on*, pages 41–50. IEEE.
- [Efthymiou et al., 2017] Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., and Christophides, V. (2017). Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In *International Semantic Web Conference*, pages 260–277. Springer.
- [Ehrlinger and Wöß, 2016] Ehrlinger, L. and Wöß, W. (2016). Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48.
- [Ellis et al., 2015] Ellis, J., Fokoue, A., Hassanzadeh, O., Kementsietsidis, A., Srinivas, K., and Ward, M. J. (2015). Exploring Big Data with Helix: Finding Needles in a Big Haystack. *SIGMOD Rec.*, 43(4):43–54.

- [Elmagarmid et al., 2006] Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. (2006). Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16.
- [Ermilov and Ngomo, 2016] Ermilov, I. and Ngomo, A.-C. N. (2016). Taipan: Automatic property mapping for tabular data. In *European Knowledge Acquisition Workshop*, pages 163–179. Springer.
- [Exner and Nugues, 2014] Exner, P. and Nugues, P. (2014). Refractive: An open source tool to extract knowledge from syntactic and semantic relations. In *LREC*, pages 2584–2589.
- [Fader et al., 2011] Fader, A., Soderland, S., and Etzioni, O. (2011). Identifying relations for open information extraction. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1535–1545. Association for Computational Linguistics.
- [Fan et al., 2010] Fan, J., Ferrucci, D., Gondek, D., and Kalyanpur, A. (2010). Prismatic: Inducing knowledge from a large scale lexicalized relation resource. In *Proceedings of the NAACL HLT 2010 first international workshop on formalisms and methodology for learning by reading*, pages 122–127. Association for Computational Linguistics.
- [Fan et al., 2014] Fan, J., Meiyu, L., Beng Chin, O., Wang-Chiew, T., and Zhang, M. (2014). A Hybrid Machine-Crowdsourcing System for Matching Web Tables. In *30th IEEE Int. Conf. on Data Engineering*, pages 976–987.
- [Fellegi and Sunter, 1969] Fellegi, I. P. and Sunter, A. B. (1969). A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210.
- [Flach and Savnik, 1999] Flach, P. A. and Savnik, I. (1999). Database dependency discovery: a machine learning approach. *AI communications*, 12(3):139–160.
- [Fossati et al., 2018] Fossati, M., Dorigatti, E., and Giuliano, C. (2018). N-ary relation extraction for simultaneous t-box and a-box knowledge base augmentation. *Semantic Web*, 9(4):413–439.
- [Freeman, 1977] Freeman, L. C. (1977). A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41.
- [Freitas et al., 2012] Freitas, A., Carvalho, D. S., da Silva, J. C. P., O’Riain, S., and Curry, E. (2012). A semantic best-effort approach for extracting structured discourse graphs from wikipedia. In *WoLE@ ISWC*, pages 70–81.
- [Furche et al., 2014] Furche, T., Gottlob, G., Grasso, G., Guo, X., Orsi, G., Schallhart, C., and Wang, C. (2014). Diadem: thousands of websites to a single database. *Proceedings of the VLDB Endowment*, 7(14):1845–1856.

- [Gal, 2011] Gal, A. (2011). *Uncertain Schema Matching*. Synthesis Lectures on Data Management. Morgan & Claypool.
- [Gal and Sagi, 2010] Gal, A. and Sagi, T. (2010). Tuning the ensemble selection process of schema matchers. *Information Systems*, 35(8):845–859.
- [Galil, 1986] Galil, Z. (1986). Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys (CSUR)*, 18(1):23–38.
- [Galindo-Legaria, 1994] Galindo-Legaria, C. A. (1994). Outerjoins as disjunctions. In *Proceedings of the ACM International Conference on Management of Data SIGMOD*, pages 348–358. ACM Press.
- [Gangemi et al., 2017] Gangemi, A., Presutti, V., Reforgiato Recupero, D., Nuzozolese, A. G., Draicchio, F., and Mongiovì, M. (2017). Semantic web machine reading with fred. *Semantic Web*, 8(6):873–893.
- [Gentile et al., 2016] Gentile, A. L., Kirstein, S., Paulheim, H., and Bizer, C. (2016). Extending rapidminer with data search and integration capabilities. In *European Semantic Web Conference*, pages 167–171. Springer.
- [Ghahramani and Heller, 2006] Ghahramani, Z. and Heller, K. A. (2006). Bayesian sets. In *Advances in neural information processing systems*, pages 435–442.
- [Gu et al., 2003] Gu, L., Baxter, R., Vickers, D., and Rainsford, C. (2003). Record linkage: Current practice and future directions. *CSIRO Mathematical and Information Sciences Technical Report*, 3:83.
- [Gulhane et al., 2011] Gulhane, P., Madaan, A., Mehta, R., Ramamirtham, J., Rastogi, R., Satpal, S., Sengamedu, S. H., Tengli, A., and Tiwari, C. (2011). Web-scale information extraction with vertex. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 1209–1220. IEEE.
- [Gupta et al., 2014] Gupta, R., Halevy, A., Wang, X., Whang, S. E., and Wu, F. (2014). Biperpedia: An ontology for search applications. *Proceedings of the VLDB Endowment*, 7(7):505–516.
- [Halevy et al., 2016] Halevy, A., Noy, N., Sarawagi, S., Whang, S. E., and Yu, X. (2016). Discovering structure in the universe of attribute names. In *Proceedings of the 25th International Conference on World Wide Web*, pages 939–949. International World Wide Web Conferences Steering Committee.
- [Hall, 1999] Hall, M. A. (1999). *Correlation-based feature selection for machine learning*. University of Waikato Hamilton, New Zealand.
- [Hao et al., 2011] Hao, Q., Cai, R., Pang, Y., and Zhang, L. (2011). From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings*

of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pages 775–784. ACM.

- [Hassanzadeh et al., 2009] Hassanzadeh, O., Chiang, F., Lee, H. C., and Miller, R. J. (2009). Framework for evaluating clustering algorithms in duplicate detection. *Proceedings of the VLDB Endowment*, 2(1):1282–1293.
- [Hassanzadeh et al., 2015] Hassanzadeh, O., Ward, M. J., Rodriguez-Muro, M., and Srinivas, K. (2015). Understanding a large corpus of web tables through matching with knowledge bases: an empirical study. In *Proc. of the 10th Int. Workshop on Ontology Matching*, pages 25–34.
- [He and Chang, 2003] He, B. and Chang, K. C.-C. (2003). Statistical schema matching across web query interfaces. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 217–228. ACM.
- [He et al., 2004] He, B., Chang, K. C.-C., and Han, J. (2004). Discovering complex matchings across web query interfaces: a correlation mining approach. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 148–157. ACM.
- [He et al., 2016] He, Y., Chakrabarti, K., Cheng, T., and Tyenda, T. (2016). Automatic discovery of attribute synonyms using query logs and table corpora. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1429–1439. International World Wide Web Conferences Steering Committee.
- [Hecht et al., 2012] Hecht, B., Carton, S. H., Quaderi, M., Schöning, J., Raubal, M., Gergle, D., and Downey, D. (2012). Explanatory semantic relatedness and explicit spatialization for exploratory search. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 415–424. ACM.
- [Hernández et al., 2015] Hernández, D., Hogan, A., and Krötzsch, M. (2015). Reifying rdf: What works well with wikidata? *SSWS@ ISWC*, 1457:32–47.
- [Hernández and Stolfo, 1998] Hernández, M. A. and Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. *Data mining and knowledge discovery*, 2(1):9–37.
- [Hignette et al., 2007] Hignette, G., Buche, P., Dibie-Barthélemy, J., and Haemmerlé, O. (2007). An ontology-driven annotation of data tables. In *International Conference on Web Information Systems Engineering*, pages 29–40. Springer.
- [Hoffart et al., 2013] Hoffart, J., Suchanek, F. M., Berberich, K., and Weikum, G. (2013). Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

- [Hogan et al., 2012] Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., and Decker, S. (2012). An empirical survey of linked data conformance. *Web Semantics: Science, Services and Agents on the World Wide Web*, 14:14–44.
- [Huhtala et al., 1999] Huhtala, Y., Kärkkäinen, J., Porkka, P., and Toivonen, H. (1999). Tane: An efficient algorithm for discovering functional and approximate dependencies. *The computer journal*, 42(2):100–111.
- [Hurst, 2001] Hurst, M. (2001). Layout and language: Challenges for table understanding on the web. In *Proceedings of the International Workshop on Web Document Analysis*, pages 27–30.
- [Isele and Bizer, 2012] Isele, R. and Bizer, C. (2012). Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649.
- [Isele and Bizer, 2013] Isele, R. and Bizer, C. (2013). Active learning of expressive linkage rules using genetic programming. *Web Semantics: Science, Services and Agents on the World Wide Web*, 23:2–15.
- [Jaccard, 1912] Jaccard, P. (1912). The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50.
- [Ji and Grishman, 2011] Ji, H. and Grishman, R. (2011). Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 1148–1158. Association for Computational Linguistics.
- [Jiménez-Ruiz and Grau, 2011] Jiménez-Ruiz, E. and Grau, B. C. (2011). Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer.
- [Jung and Kwon, 2006] Jung, S.-W. and Kwon, H.-C. (2006). A scalable hybrid approach for extracting head components from web tables. *IEEE Transactions on Knowledge and Data Engineering*, 18(2):174–187.
- [Kacprzak et al., 2017] Kacprzak, E., Koesten, L. M., Ibáñez, L.-D., Simperl, E., and Tennison, J. (2017). A query log analysis of dataset search. In *International Conference on Web Engineering*, pages 429–436. Springer.
- [Kim and Lee, 2005] Kim, Y.-S. and Lee, K.-H. (2005). Detecting tables in web documents. *Engineering Applications of Artificial Intelligence*, 18(6):745–757.
- [Kleppmann et al., 2018] Kleppmann, B., Bizer, C., Yaqub, E., Temme, F., Schlunder, P., Arnau, D., and Klinkenberg, R. (2018). Density-and correlation-based table extension. In *CEUR Workshop Proceedings*, volume 2191, pages 191–194. RWTH.

- [Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.
- [Köpcke and Rahm, 2010] Köpcke, H. and Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210.
- [Kushmerick et al., 1997] Kushmerick, N., Weld, D. S., and Doorenbos, R. (1997). *Wrapper induction for information extraction*. University of Washington Washington.
- [Lautert et al., 2013] Lautert, L. R., Scheidt, M. M., and Dorneles, C. F. (2013). Web table taxonomy and formalization. *ACM SIGMOD Record*, 42(3):28–33.
- [Lee et al., 2013] Lee, T., Wang, Z., Wang, H., and Hwang, S.-w. (2013). Attribute extraction and scoring: A probabilistic approach. In *29th International Conference on Data Engineering (ICDE)*, pages 194–205.
- [Lee et al., 2007] Lee, Y., Sayyadian, M., Doan, A., and Rosenthal, A. S. (2007). eTuner: tuning schema matching software using synthetic scenarios. *The VLDB Journal – The International Journal on Very Large Data Bases*, 16(1):97–122.
- [Lehmann et al., 2015] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C. (2015). DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6(2):167–195.
- [Lehmberg and Bizer, 2016] Lehmberg, O. and Bizer, C. (2016). Web table column categorisation and profiling. In *Proceedings of the 19th International Workshop on Web and Databases*, page 4. ACM.
- [Lehmberg and Bizer, 2017] Lehmberg, O. and Bizer, C. (2017). Stitching web tables for improving matching quality. *Proceedings of the VLDB Endowment*, 10(11):1502–1513.
- [Lehmberg and Bizer, 2019a] Lehmberg, O. and Bizer, C. (2019a). Profiling the semantics of n-ary web table data. In *Proceedings of the International Workshop on Semantic Big Data, SBD ’19*, pages 5:1–5:6, New York, NY, USA. ACM.
- [Lehmberg and Bizer, 2019b] Lehmberg, O. and Bizer, C. (2019b). Synthesizing n-ary relations from web tables. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics, WIMS2019*, pages 17:1–17:12, New York, NY, USA. ACM.
- [Lehmberg et al., 2017] Lehmberg, O., Brinkmann, A., and Bizer, C. (2017). Winte.r – a web data integration framework. In *CEUR workshop proceedings*, volume 1963, pages Paper–506. RWTH.

- [Lehmberg and Hassanzadeh, 2018] Lehmberg, O. and Hassanzadeh, O. (2018). Ontology augmentation through matching with web tables. In *Ontology Matching: OM-2018: Proceedings of the ISWC Workshop*, page 37.
- [Lehmberg et al., 2016] Lehmberg, O., Ritze, D., Meusel, R., and Bizer, C. (2016). A large public corpus of web tables containing time and context meta-data. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76. International World Wide Web Conferences Steering Committee.
- [Lehmberg et al., 2015] Lehmberg, O., Ritze, D., Ristoski, P., Meusel, R., Paulheim, H., and Bizer, C. (2015). The Mannheim Search Join Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:159–166.
- [Levene and Loizou, 1998] Levene, M. and Loizou, G. (1998). Axiomatisation of functional dependencies in incomplete relations. *Theoretical Computer Science*, 206(1):283–300.
- [Levenshtein, 1966] Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.
- [Lim and Ng, 1999] Lim, S.-J. and Ng, Y.-K. (1999). An automated approach for retrieving hierarchical data from html tables. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 466–474. ACM.
- [Limaye et al., 2010] Limaye, G., Sarawagi, S., and Chakrabarti, S. (2010). Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc.of the VLDB Endow.*, 3:1338–1347.
- [Ling et al., 2013] Ling, X., Halevy, A. Y., Wu, F., and Yu, C. (2013). Synthesizing union tables from the web. In *Twenty-Third International Joint Conference on Artificial Intelligence*.
- [Liu et al., 2010] Liu, J., Li, J., Liu, C., and Chen, Y. (2010). Discover dependencies from data – a review. *IEEE Transactions on Knowledge and Data Engineering*, 24(2):251–264.
- [Lu and Getoor, 2003] Lu, Q. and Getoor, L. (2003). Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 496–503.
- [Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.
- [Mannino et al., 1988] Mannino, M. V., Chu, P., and Sager, T. (1988). Statistical profile estimation in database systems. *ACM Computing Surveys (CSUR)*, 20(3):191–221.

- [Manola et al., 2004] Manola, F., Miller, E., McBride, B., et al. (2004). Rdf primer. *W3C recommendation*, 10(1-107):6.
- [Meusel et al., 2014] Meusel, R., Vigna, S., Lehmberg, O., and Bizer, C. (2014). Graph structure in the web—revisited: a trick of the heavy tail. In *Proceedings of the 23rd international conference on World Wide Web*, pages 427–432. ACM.
- [Meusel et al., 2015] Meusel, R., Vigna, S., Lehmberg, O., and Bizer, C. (2015). The graph structure in the web: Analyzed on different aggregation levels. *The Journal of Web Science*, 1(1):33–47.
- [Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [Miller, 2018] Miller, R. J. (2018). Open data integration. *Proceedings of the VLDB Endowment*, 11(12):2130–2139.
- [Miller et al., 2000] Miller, R. J., Haas, L. M., and Hernández, M. A. (2000). Schema mapping as query discovery. In *VLDB*, volume 2000, pages 77–88.
- [Mulwad et al., 2013] Mulwad, V., Finin, T., and Joshi, A. (2013). Semantic message passing for generating linked data from tables. In *International Semantic Web Conference*, pages 363–378. Springer.
- [Munoz et al., 2013] Munoz, E., Hogan, A., and Mileo, A. (2013). Triplifying wikipedia’s tables. *LD4IE@ISWC*, 100(101):102.
- [Naumann, 2014] Naumann, F. (2014). Data profiling revisited. *ACM SIGMOD Record*, 42(4):40–49.
- [Navarro, 2001] Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.
- [Newcombe et al., 1959] Newcombe, H. B., Kennedy, J. M., Axford, S., and James, A. P. (1959). Automatic linkage of vital records. *Science*, 130(3381):954–959.
- [Nishida et al., 2017] Nishida, K., Sadamitsu, K., Higashinaka, R., and Matsuo, Y. (2017). Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*.
- [On et al., 2007] On, B.-W., Koudas, N., Lee, D., and Srivastava, D. (2007). Group linkage. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 496–505. IEEE.

- [Oulabi and Bizer, 2017] Oulabi, Y. and Bizer, C. (2017). Estimating missing temporal meta-information using knowledge-based-trust. In *KDWEB 2017 : proceedings of the 3rd International Workshop on Knowledge Discovery on the WEB Cagliari, Italy, September 11 to 12, 2017*, volume 1959, page Paper 4, Aachen. RWTH.
- [Oulabi and Bizer, 2019] Oulabi, Y. and Bizer, C. (2019). Extending cross-domain knowledge bases with long tail entities using web table data. In *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019 : proceedings*, pages 385–396, Konstanz. OpenProceedings.org. Online-Ressource.
- [Oulabi et al., 2016] Oulabi, Y., Meusel, R., and Bizer, C. (2016). Fusing time-dependent web table data. In *Proceedings of the 19th International Workshop on Web and Databases*, page 3. ACM.
- [Özsu and Valduriez, 2011] Özsu, M. T. and Valduriez, P. (2011). *Principles of distributed database systems*. Springer Science & Business Media.
- [Page et al., 1999] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: bringing order to the Web. Technical report, Stanford InfoLab.
- [Papenbrock et al., 2015] Papenbrock, T., Ehrlich, J., Marten, J., Neubert, T., Rudolph, J.-P., Schönberg, M., Zwiener, J., and Naumann, F. (2015). Functional dependency discovery: An experimental evaluation of seven algorithms. *Proceedings of the VLDB Endowment*, 8(10):1082–1093.
- [Papenbrock and Naumann, 2016] Papenbrock, T. and Naumann, F. (2016). A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data*, pages 821–833. ACM.
- [Papenbrock and Naumann, 2017] Papenbrock, T. and Naumann, F. (2017). Data-driven schema normalization. In *EDBT*, pages 342–353.
- [Paulheim, 2017] Paulheim, H. (2017). Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508.
- [Penn et al., 2001] Penn, G., Hu, J., Luo, H., and McDonald, R. (2001). Flexible web document analysis for delivery to narrow-bandwidth devices. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 1074–1078. IEEE.
- [Pham et al., 2016] Pham, M., Alse, S., Knoblock, C. A., and Szekely, P. (2016). Semantic labeling: a domain-independent approach. In *International Semantic Web Conference*, pages 446–462. Springer.

- [Pimplikar and Sarawagi, 2012] Pimplikar, R. and Sarawagi, S. (2012). Answering Table Queries on the Web Using Column Keywords. *Proc. VLDB Endow.*, 5(10):908–919.
- [Pinto et al., 2002] Pinto, D., Branstein, M., Coleman, R., Croft, W. B., King, M., Li, W., and Wei, X. (2002). Quasm: a system for question answering using semi-structured data. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 46–55. ACM.
- [Poosala et al., 1996] Poosala, V., Haas, P. J., Ioannidis, Y. E., and Shekita, E. J. (1996). Improved histograms for selectivity estimation of range predicates. In *ACM Sigmod Record*, volume 25, pages 294–305. ACM.
- [Qiu et al., 2015] Qiu, D., Barbosa, L., Dong, X. L., Shen, Y., and Srivastava, D. (2015). Dexter: large-scale discovery and extraction of product specifications on the web. *Proceedings of the VLDB Endowment*, 8(13):2194–2205.
- [Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350.
- [Rahm and Do, 2000] Rahm, E. and Do, H. H. (2000). Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13.
- [Raman and Hellerstein, 2001] Raman, V. and Hellerstein, J. M. (2001). Potter’s wheel: An interactive data cleaning system. In *VLDB*, volume 1, pages 381–390.
- [Rao and Zhu, 2016] Rao, B. and Zhu, E. (2016). Searching web data using min-hash lsh. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2257–2258. ACM.
- [Ravi and Paşca, 2008] Ravi, S. and Paşca, M. (2008). Using structured text for large-scale attribute extraction. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1183–1192. ACM.
- [Rinser et al., 2013] Rinser, D., Lange, D., and Naumann, F. (2013). Cross-Lingual Entity Matching and Infobox Alignment in Wikipedia. *Inf. Syst.*, 38:887–907.
- [Ritze, 2017] Ritze, D. (2017). *Web-scale web table to knowledge base matching*. PhD thesis.
- [Ritze and Bizer, 2017] Ritze, D. and Bizer, C. (2017). Matching web tables to dbpedia-a feature utility study. *Proceedings of the 20th International Conference on Extending Database Technology*, 42(41):19.
- [Ritze et al., 2015] Ritze, D., Lehmborg, O., and Bizer, C. (2015). Matching HTML Tables to DBpedia. In *Proc. of the 5th Int. Conference on Web Intelligence, Mining and Semantics*.

- [Ritze et al., 2016] Ritze, D., Lehmberg, O., Oulabi, Y., and Bizer, C. (2016). Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *Proceedings of the 25th International Conference on World Wide Web*, pages 251–261. International World Wide Web Conferences Steering Committee.
- [Saleiro et al., 2017] Saleiro, P., Milic-Frayling, N., Mendes Rodrigues, E., and Soares, C. (2017). Relink: A research framework and test collection for entity-relationship retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1273–1276. ACM.
- [Sarawagi and Chakrabarti, 2014] Sarawagi, S. and Chakrabarti, S. (2014). Open-domain quantity queries on web tables: annotation, response, and consensus models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 711–720. ACM.
- [Sarawagi and Kirpal, 2004] Sarawagi, S. and Kirpal, A. (2004). Efficient set joins on similarity predicates. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 743–754. ACM.
- [Schmachtenberg et al., 2014] Schmachtenberg, M., Bizer, C., and Paulheim, H. (2014). Adoption of the Linked Data Best Practices in Different Topical Domains. In *Proc. of the 13th Int. Semantic Web Conference (ISWC14)*, volume 8796, pages 245–260. Springer International Publishing.
- [Schmitz et al., 2012] Schmitz, M., Bart, R., Soderland, S., Etzioni, O., et al. (2012). Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534. Association for Computational Linguistics.
- [Sekhavat et al., 2014] Sekhavat, Y. A., di Paolo, F., Barbosa, D., and Merialdo, P. (2014). Knowledge Base Augmentation using Tabular Data. In *Proc. of the 7th Workshop on Linked Data on the Web*.
- [Singhal, 2012] Singhal, A. (2012). Introducing the knowledge graph: Things, not string. Blog. Retrieved March 19, 2015.
- [Son and Park, 2013] Son, J.-W. and Park, S.-B. (2013). Web table discrimination with composition of rich structural and content information. *Applied Soft Computing*, 13(1):47–57.
- [Song et al., 2015] Song, W., Zhao, S., Zhang, C., Wu, H., Wang, H., Liu, L., and Wang, H. (2015). Exploiting collective hidden structures in webpage titles for open domain entity extraction. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1014–1024. International World Wide Web Conferences Steering Committee.

- [Su et al., 2006] Su, W., Wang, J., and Lochovsky, F. (2006). Holistic Schema Matching for Web Query Interfaces. In Ioannidis, Y., Scholl, M., Schmidt, J., Matthes, F., Hatzopoulos, M., Boehm, K., Kemper, A., Grust, T., and Boehm, C., editors, *Advances in Database Technology - EDBT 2006*, volume 3896 of *Lecture Notes in Computer Science*, pages 77–94. Springer Berlin Heidelberg.
- [Suchanek et al., 2011] Suchanek, F., Abiteboul, S., and Senellart, P. (2011). Paris: Probabilistic alignment of Relations, Instances, and Schema. *Proc. VLDB Endowment*, 5:157–168.
- [Suchanek et al., 2007] Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.
- [Sun et al., 2016] Sun, H., Ma, H., He, X., Yih, W.-t., Su, Y., and Yan, X. (2016). Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*, pages 771–782. International World Wide Web Conferences Steering Committee.
- [Surdeanu and Ji, 2014] Surdeanu, M. and Ji, H. (2014). Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population Evaluation. <http://nlp.cs.rpi.edu/paper/sf2014overview.pdf>.
- [Tao and Embley, 2009] Tao, C. and Embley, D. W. (2009). Automatic hidden-web table interpretation, conceptualization, and semantic annotation. *Data & Knowledge Engineering*, 68(7):683–703.
- [Ukkonen, 1992] Ukkonen, E. (1992). Approximate string-matching with q-grams and maximal matches. *Theoretical computer science*, 92(1):191–211.
- [Ullman, 1989] Ullman, J. D. (1989). Principles of database and knowledge-base systems. *Computer Science Press, New York*.
- [Ullmann, 1977] Ullmann, J. R. (1977). A binary n-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words. *The Computer Journal*, 20(2):141–147.
- [Usbeck et al., 2014] Usbeck, R., Ngomo, A.-C. N., Röder, M., Gerber, D., Coelho, S. A., Auer, S., and Both, A. (2014). Agdistis-graph-based disambiguation of named entities using linked data. In *International semantic web conference*, pages 457–471. Springer.
- [Venetis et al., 2011] Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., and Wu, C. (2011). Recovering Semantics of Tables on the Web. *Proc. of the VLDB Endow.*, pages 528–538.
- [Vrandečić and Krötzsch, 2014] Vrandečić, D. and Krötzsch, M. (2014). Wiki-data: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

- [(W3C), 2014] (W3C), W. W. W. C. (2014). Rdf 1.1 concepts and abstract syntax. <https://www.w3.org/TR/rdf11-concepts/>.
- [Wang et al., 2015a] Wang, C., Chakrabarti, K., He, Y., Ganjam, K., Chen, Z., and Bernstein, P. A. (2015a). Concept expansion using web tables. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1198–1208. International World Wide Web Conferences Steering Committee.
- [Wang et al., 2009] Wang, D. Z., Dong, X. L., Sarma, A. D., Franklin, M. J., and Halevy, A. Y. (2009). Functional dependency generation and applications in pay-as-you-go data integration systems. In *12th International Workshop on the Web and Databases (WebDB)*.
- [Wang et al., 2015b] Wang, H., Liu, A., Wang, J., Ziebart, B. D., Yu, C. T., and Shen, W. (2015b). Context retrieval for web tables. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, pages 251–260. ACM.
- [Wang et al., 2000] Wang, H.-L., Wu, S.-H., Wang, I., Sung, C.-L., Hsu, W.-L., and Shih, W.-K. (2000). Semantic search on internet tabular information extraction for answering queries. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 243–249. ACM.
- [Wang and Madnick, 1989] Wang, J. and Madnick, S. E. (1989). The inter-database instance identification problem in integrating autonomous systems. In *[1989] Proceedings. Fifth International Conference on Data Engineering*, pages 46–55. IEEE.
- [Wang et al., 2012] Wang, J., Wang, H., Wang, Z., and Zhu, K. Q. (2012). Understanding Tables on the Web. In *Proc. of the 31st Int. Conf. on Conceptual Modeling*, pages 141–155.
- [Wang et al., 2004] Wang, J., Wen, J.-R., Lochovsky, F., and Ma, W.-Y. (2004). Instance-based schema matching for web databases by domain-specific query probing. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 408–419. VLDB Endowment.
- [Wang and Cohen, 2008] Wang, R. C. and Cohen, W. W. (2008). Iterative set expansion of named entities using the web. In *Proc. of the 8th IEEE Int. Conference on Data Mining, ICDM '08*, pages 1091–1096.
- [Wang and Strong, 1996] Wang, R. Y. and Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of management information systems*, 12(4):5–33.
- [Wang and He, 2017] Wang, Y. and He, Y. (2017). Synthesizing mapping relationships using table corpus. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1117–1132. ACM.

- [Wang and Hu, 2002] Wang, Y. and Hu, J. (2002). Detecting Tables in HTML Documents. In *Proc. of the 5th Int. Workshop on Document Analysis Systems V*, pages 249–260.
- [Weikum and Theobald, 2010] Weikum, G. and Theobald, M. (2010). From Information to Knowledge: Harvesting Entities and Relationships from Web Sources. In *Proc. of the 29th Symp. on Principles of Database Systems*, pages 65–76.
- [Winkler, 2006] Winkler, W. E. (2006). Overview of record linkage and current research directions. In *Bureau of the Census*. Citeseer.
- [Witten and Milne, 2008] Witten, I. H. and Milne, D. N. (2008). An effective, low-cost measure of semantic relatedness obtained from wikipedia links. pages 25–30. AAAI Press.
- [Yakout et al., 2012] Yakout, M., Ganjam, K., Chakrabarti, K., and Chaudhuri, S. (2012). InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. In *Proc. of the 2012 SIGMOD*, pages 97–108.
- [Yang, 1991] Yang, W. (1991). Identifying syntactic differences between two programs. *Software: Practice and Experience*, 21(7):739–755.
- [Yao and Hamilton, 2008] Yao, H. and Hamilton, H. J. (2008). Mining functional dependencies from data. *Data Mining and Knowledge Discovery*, 16(2):197–219.
- [Yates et al., 2007] Yates, A., Cafarella, M., Banko, M., Etzioni, O., Broadhead, M., and Soderland, S. (2007). Texrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics.
- [Yin and Tan, 2011] Yin, X. and Tan, W. (2011). Semi-supervised truth discovery. In *Proc. of the 20th Int. Conference on World Wide Web, WWW '11*, pages 217–226. AC.
- [Yin et al., 2010] Yin, X., Tan, W., Li, X., and Tu, Y.-C. (2010). Automatic extraction of clickable structured web contents for name entity queries. In *Proceedings of the 19th international conference on World wide web*, pages 991–1000. ACM.
- [Yin et al., 2011] Yin, X., Tan, W., and Liu, C. (2011). FACTO: A Fact Lookup Engine Based on Web Tables. In *Proceedings of the 20th International Conference on World Wide Web, WWW '11*, pages 507–516, New York, NY, USA. ACM.

- [Yoshida et al., 2001] Yoshida, M., Torisawa, K., and Tsujii, J. (2001). A method to integrate tables of the world wide web. In *Proceedings of the International Workshop on Web Document Analysis (WDA 2001)*, pages 31–34.
- [Zanibbi et al., 2004] Zanibbi, R., Blostein, D., and Cordy, J. R. (2004). A survey of table recognition. *Document Analysis and Recognition*, 7(1):1–16.
- [Zhang and Lee, 2003] Zhang, D. and Lee, W. S. (2003). Question classification using support vector machines. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 26–32. ACM.
- [Zhang and Chakrabarti, 2013] Zhang, M. and Chakrabarti, K. (2013). Info-Gather+: Semantic Matching and Annotation of Numeric and Time-varying Attributes in Web Tables. In *Proc. of the 2013 ACM SIGMOD Int. Conference on Management of Data*, pages 145–156.
- [Zhang et al., 2013] Zhang, X., Chen, Y., Chen, J., Du, X., and Zou, L. (2013). Mapping Entity-Attribute Web Tables to Web-Scale Knowledge Bases. In *Database Systems for Advanced Applications*, pages 108–122. Springer Berlin.
- [Zhang, 2017] Zhang, Z. (2017). Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, 8(6):921–957.
- [Zhou et al., 2007] Zhou, X., Gaugaz, J., Balke, W.-T., and Nejdl, W. (2007). Query relaxation using malleable schemas. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 545–556. ACM.
- [Zhu et al., 2019] Zhu, E., Deng, D., Nargesian, F., and Miller, R. J. (2019). Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*, pages 847–864. ACM.
- [Zhu et al., 2016] Zhu, E., Nargesian, F., Pu, K. Q., and Miller, R. J. (2016). Lsh ensemble: Internet-scale domain search. *Proceedings of the VLDB Endowment*, 9(12):1185–1196.
- [Zhu et al., 2017] Zhu, E., Pu, K. Q., Nargesian, F., and Miller, R. J. (2017). Interactive navigation of open data linkages. *Proceedings of the VLDB Endowment*, 10(12):1837–1840.
- [Zwicklbauer et al., 2016] Zwicklbauer, S., Seifert, C., and Granitzer, M. (2016). Doser-a knowledge-base-agnostic framework for entity disambiguation using semantic embeddings. In *International Semantic Web Conference*, pages 182–198. Springer.